[redacted]

1 / 1

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-236249
(43)Date of publication of application : 29.08.2000

(51)Int.Cl. H03K 19/173
G01R 31/28
G06F 17/50
H01L 21/82
H03K 19/177
// G06F 11/22

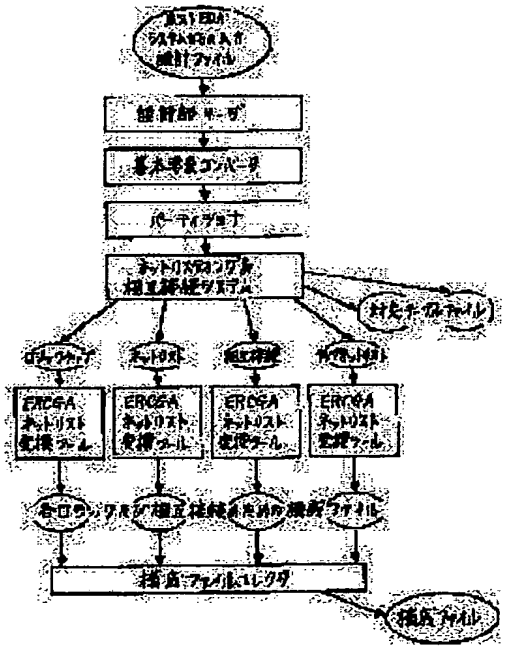
(21)Application number : 11-336488
(22)Date of filing : 04.10.1989

(71)Applicant : QUICKTURN DESIGN SYST INC
(72)Inventor : BUTTS MICHAEL R
BATCHELLER JON A

(30)Priority
Priority number : 88 254463 Priority date : 05.10.1988 Priority country : US

(54) METHOD FOR STRUCTURING LOGICAL CONFIGURATION USING ELECTRICALLY RECONFIGURABLE GATE ARRAY

(57)Abstract:
PROBLEM TO BE SOLVED: To actualize the method for structuring a logical configuration by using the electrically reconfigurable gate array.
SOLUTION: Electrically reconfigurable gate array (ERCGA) logic chips are connected to one another via reconfigurable interconnections. The electrical representation of a large-scale digital network is so converted as to adopt a hardware style, which operates actually and temporarily on the interconnected chips. The digital network actualized on the interconnected chips can be changed any time through reconfigured connections. Consequently, a system is adapted to various purposes which include simulation, prototyping, execution, and computation. The reconfigurable interconnections are configured by the ERCGA chips dedicated to an interconnected function. Each interconnected ERCGA is not connected to all the interconnected chips but is connected to at least one pin.



LEGAL STATUS
[Date of request for examination] 27.12.1999

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－236249
（P2000－236249A）
(43)公開日　平成12年８月29日(2000.8.29)

(51)Int.Cl.'	識別記号	ＦＩ		テーマコード'（参考）
Ｈ０３Ｋ　19/173	１０１	Ｈ０３Ｋ	19/173	１０１
Ｇ０１Ｒ　31/28			19/177	
Ｇ０６Ｆ　17/50		Ｇ０６Ｆ	11/22	３３０Ｄ
Ｈ０１Ｌ　21/82		Ｇ０１Ｒ	31/28	Ｆ
Ｈ０３Ｋ　19/177		Ｇ０６Ｆ	15/60	６６４Ｐ

審査請求　有　請求項の数４　ＯＬ　（全 81 頁）　最終頁に続く

(21)出願番号　特願平11－336488
(62)分割の表示　特願平1－509588の分割
(22)出願日　平成１年10月４日(1989.10.4)

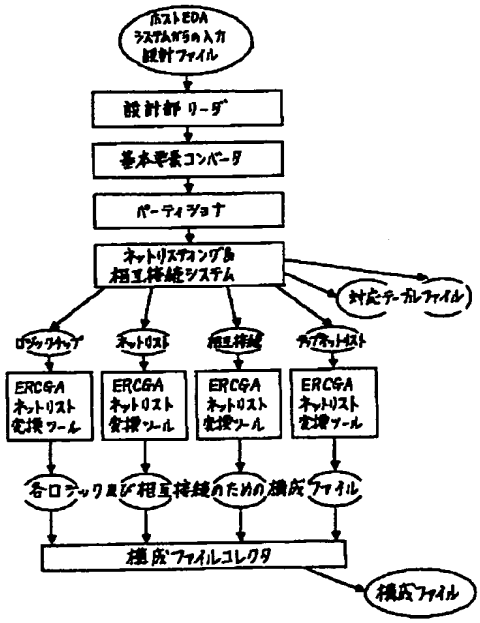
(31)優先権主張番号　２５４４６３
(32)優先日　昭和63年10月５日(1988.10.5)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　595043376
クイックターン・デザイン・システムズ・インコーポレイテッド
ＱＵＩＣＫＴＵＲＮ　ＤＥＳＩＧＮ　ＳＹＳＴＥＭＳ，　ＩＮＣ．
アメリカ合衆国95131カリフォルニア州サンノゼ、ウエスト・トリンプル・ロード55番
(72)発明者　パツツ　マイケル　アール
アメリカ合衆国オレゴン州　97212　ポートランド　エヌ　イー　アラミーダ　3336
(74)代理人　100064344
弁理士　岡田　英彦　（外３名）

最終頁に続く

(54)【発明の名称】　電気的に再構成可能なゲートアレイを用いて論理構成を構築する方法

(57)【要約】
【課題】　電気的に再構成可能なゲートアレイを用いて論理構成を構築する方法を提供する。
【解決手段】　複数の電気的に再構成可能なゲートアレイ（ＥＲＣＧＡ）論理チップは、再構成可能な相互接続を介して相互に接続されている。そして、大規模デジタル回路網の電気的表現は、相互接続チップ上で一時的に実際に動作するハードウェア形態を採るように変換される。再構成接続により、相互接続チップ上に実現されたデジタル回路網は随時変更される。これによって、システムは、シミュレーション、プロトタイピング、実行、計算を含む種々の目的に適合される。再構成可能な相互接続は、相互接続機能専用のＥＲＣＧＡチップにより構成されている。各相互接続ＥＲＣＧＡは、複数の相互接続チップの全てではないが少なくとも一つのピンに接続されている。



【特許請求の範囲】

【請求項１】　第１及び第２の電気的に再構成可能なゲートアレイ（ＥＲＣＧＡ）を設ける工程と；ブールのロジックゲートから成る基本要素を具え、第１デジタルロジック回路網を表している第１入力データ及び、前記基本要素を相互接続する回路網を設ける工程と；前記第１入力データを、第１及び第２部分に分割する工程と；分割された第１データの第１部分を、第１ＥＲＣＧＡに供給し、これによって表現される前記第１ロジック回路網の第１部分が、第１ＥＲＣＧＡにおいて実際に動作形態を採るようにする工程と；前記分割された第１データの第２部分を、第２ＥＲＣＧＡに供給し、これによって表現される前記第１デジタルロジック回路網の第２部分が、第２ＥＲＣＧＡにおいて実際に動作形態を採るようにする工程と；前記第１及び第２ＥＲＣＧＡを相互接続し、前記第１入力データで特定される少なくとも一個の回路網が、前記第１及び第２ＥＲＣＧＡの間に及ぶようにする工程と；ブールのロジックゲートから成る基本要素と、前記基本要素を相互接続する回路網とを具えていることを除き、前記第１デジタルロジック回路網と全く無関係な、第２デジタルロジック回路網を表現している第２入力データを供給し、前記第１及び第２デジタルロジック回路網が同一のＥＲＣＧＡにおいて実際の動作形態を採るようにする工程と；前記第２入力データを、第１及び第２部分に分割する工程と；分割された第２データの第１部分を、第１ＥＲＣＧＡに供給し、これによって表現される前記第２デジタルロジック回路網の第１部分が、前記第１ＥＲＣＧＡにおいて実際に動作形態を採ることができるようにする工程と；前記分割された第２データの第２部分を、第２ＥＲＣＧＡに供給し、これによって表現される前記第２デジタルロジック回路網の第２部分が、前記第２ＥＲＣＧＡで実際に動作形態を採ることができるようにする工程と；前記第１及び第２ＥＲＣＧＡを相互接続し、前記第２入力データで特定される少なくとも１個の回路網が、前記第１及び第２ＥＲＣＧＡの間に及ぶように構成する工程；とを具えることを特徴とする方法。

【請求項２】　前記区分化の工程を、自動的に行うことを特徴とする請求項１に記載の方法。

【請求項３】　シミュレートされる第１デジタルロジック回路網を規定する工程と；前記第１デジタルロジック回路網を表現している第１入力データを発生させる工程と；前記第１入力データを、第１及び第２部分に分割する工程と；前記分割された第１データの前記第１部分を、前記第１ＥＲＣＧＡに供給し、このようにして表現される前記第１デジタルロジック回路網の第１部分が、前記第１ＥＲＣＧＡにおいて、実際に動作する形態を採ることができるようにする工程と；前記分割された第１データの前記第２部分を前記第２ＥＲＣＧＡに供給し、このようにして表現される前記第１ロジック回路網の第２部分が、前記第２ＥＲＣＧＡにおいて実際に動作する形態を採ることができるようにする工程と；前記第１及び第２ＥＲＣＧＡを相互接続し、前記第１入力データで特定される少なくとも１個の回路網が、前記第１及び第２ＥＲＣＧＡの間に及ぶようにする工程と；第１シミュレーションで使用する一組の第１刺激をソフトウェアで規定する工程と；前記刺激を規定するソフトウェアを、第１電気信号に変換する工程と；前記入力信号としての第１電気信号を、相互接続された前記第１及び第２ＥＲＣＧＡに供給する工程と；相互接続された前記第１及び第２ＥＲＣＧＡから、第１出力電気信号を受信する工程と；前記第１電気出力信号を、ソフトウェアの形態に変換する工程；とを具え、且つ、前記第１デジタルロジック回路網とは別の第２デジタルロジック回路網において、上記工程を繰り返す工程；を具えている請求項１に記載のシミュレーション方法。

【請求項４】　合成ツールを使用して、第１コンピュータプログラムを、該第１コンピュータプログラムで表現されるアルゴリズムに従って動作する第１デジタルロジック回路網を表現している一組の第１入力データに変換する工程と；前記第１入力データを、前記第１及び第２部分に分割する工程と；前記分割された第１データの前記第１部分を、前記第１ＥＲＣＧＡに供給し、このようにして表現される前記第１デジタルロジック回路網の第１部分が、前記第１ＥＲＣＧＡにおて、実際の動作形態を採ることができるようにする工程と；前記分割された第１データの前記第２部分を前記第２ＥＲＣＧＡに供給し、このようにして表現される前記第１デジタルロジック回路網の第２部分が、前記第２ＥＲＣＧＡにおいて、実際の動作形態を採ることができるようにする工程と；前記第１及び第２ＥＲＣＧＡを相互接続し、前記第１入力データが特定する少なくとも一個の回路網が、前記第１及び第２ＥＲＣＧＡの間に及ぶようにする工程と；前記第１プログラムの入力データに対応する第１刺激信号を発生させる工程と；前記第１刺激信号を、入力信号として、相互接続された前記第１及び第２ＥＲＣＧＡに供給する工程と；相互接続された、前記第１及び第２ＥＲＣＧＡから、前記第１プログラムの出力データに対応する第１出力電気信号を受信する工程；とを具え、且つ、前記第１デジタル回路網とは別の第２デジタル回路網において、上記工程を繰り返す工程；を具えている請求項１に記載の計算方法。

【請求項５】　前記合成ツール使用工程が：設計部合成ツールを使用して、前記第１コンピュータプログラムを、データパス及び有限状態マシンコントローラから成り、前記第１プログラムによって表現されるアルゴリズムに従って作動するシステムの表現に変換する工程と；ロジック合成ツールを用いて、前記設計部合成ツールによって提供されるデータパス及び有限状態マシンコントローラの表現を一組の第１入力データに変換する工程；

とを具えることを特徴とする請求項１に記載の方法。
【請求項６】　前記ＥＲＣＧＡが、各々、複数のピンを具え、且つ、前記相互接続の工程が：少なくとも一個の追加のＥＲＣＧＡを設け、再構成可能な相互接続としての役割を果たすようにする工程と；前記再構成可能な相互接続ＥＲＣＧＡの各々を、前記第１及び第２ＥＲＣＧＡのピンのすべてではないが、少なくとも一個に接続する工程；とを具えていることを特徴とする請求項１に記載の方法。
【請求項７】　(a)Ｎ個のＥＲＣＧＡを設ける工程と；
(b)前記第１入力データを、Ｎ個の部分に分割する工程と；
(c)分割されたデータの各部分を、対応するＥＲＣＧＡに供給し、このようにして表現される前記デジタルロジック回路網の前記部分が、前記ＥＲＣＧＡにおいて実際の動作形態を採ることができるようにしている工程と；
(d)Ｎ個のＥＲＣＧＡを相互接続し、ＥＲＣＧＡの各々を、少なくとも一個の他のＥＲＣＧＡに接続し、且つ、前記入力データで特定される回路網の各々を実現する工程と；
(e)前記第２入力データに対して、(b)、(c)及び(d)の工程を繰り返す工程；とを更に具えていることを特徴とする請求項１に記載の方法。
【請求項８】　ＥＲＣＧＡが各々複数のピンを具え、且つ、前記相互接続の工程が：少なくとも一個の追加のＥＲＣＧＡを設け、再構成可能な相互接続として役割を果たすようにする工程と；前記再構成可能な相互接続ＥＲＣＧＡの各々を、前記複数であるＮ個のＥＲＣＧＡのピンのすべてではないが、少なくとも個に接続する工程；とを具えていることを特徴とする請求項７に記載の方法。
【請求項９】　前記再構成可能な相互接続ＥＲＣＧＡを、前記Ｎ個の各々のＥＲＣＧＡの全てではないが、少なくとも一個に接続する工程を、更に具えていることを特徴とするむ請求項８に記載の方法。
【請求項１０】　順次、というよりはむしろ単一のプロセスで：前記入力データを分割する工程と；これによって、対応して必要となる相互接続の特徴を識別する工程；とを実行する工程を更に具え、これによって、前記入力データを、このような方法で分割し、対応して必要な相互接続を簡易化することを特徴とする請求項９に記載の方法。
【請求項１１】　シード基本要素を決定し、他の基本要素をこれに加えることによって前記入力データを分割し、これによって、基本要素のクラスタを構成する工程を更に具え；前記基本要素の各々が、多数のピンを具え；前記クラスタの構成が、クラスタに割り当てられていない各基本要素の有効な機能を評価する工程を具え；前記有効な機能が、最も多数のピンを有する基本要素に、最大の初期利益をもたらすことを特徴とする請求項１０に記載の方法。
【請求項１２】　シード基本要素を決定するとともに、他の基本要素をこっちに加えることによって、前記入力データを分割し、これによって、基本要素のクラスタを構成する工程と；相互接続の限界に至るまで、基本要素をクラスタから取り除く工程とを更に具え；前記分割の工程が、相互接続の限界を超えて、基本要素をクラスタに加える方法を備えていることを特徴とする請求項１０に記載の方法。
【請求項１３】　前記相互接続の工程が、更に：２個のＥＲＣＧＡ間に及んでいる回路網が：回路網のルートを決定することのできる複数の対象となる再構成可能な相互接続ＥＲＣＧＡを試験する工程と；少なくとも部分的に、ＥＲＣＧＡがすでに用いられている利用の程度に基づき、このような相互接続ＥＲＣＧＡの各々を介してのルート決定の適性を評価する工程；とを具えていることを特徴とする請求項１０に記載の方法。
【請求項１４】　(a)Ｎ個のＥＲＣＧＡを設ける工程と；
(b)回路形態的に、前記Ｎ個のＥＲＣＧＡを、規則的な多次元アレイに配置し、これによって相対的に隣接するＥＲＣＧＡを決定する工程と；
(c)直接的に隣接するＥＲＣＧＡを相互接続する工程と；
(d)前記第１入力データを、Ｎ個の部分に分割する工程と；
(e)前記分割されたデータの各部分を、対応するＥＲＣＧＡに供給し、このようにして表現される前記デジタルロジック回路網の前記部分が、前記ＥＲＣＧＡにおいて実際に動作する形態を採ることができるようにする工程と；
(f)非隣接ＥＲＣＧＡ間に介在するＥＲＣＧＡを介して相互接続を確立することで、非隣接ＥＲＣＧＡを相互接続し、必要されるＮ個のＥＲＣＧＡを相互接続し、前記第１データで特定される回路網を実現する工程と；
(g)前記第２入力データに対して、工程(d)、(e)及び(f)を繰り返す工程；とを更に具えていることを特徴とする請求項１に記載の方法。
【請求項１５】　自動ルーティング方法を用いて、非隣接ＥＲＣＧＡを相互接続するのに、介在するどのＥＲＣＧＡ及びピンを用いるかを決定する工程を更に具えていることを特徴とする請求項１４に記載の方法。
【請求項１６】　前記デジタルロジック回路網中の故障を、前記入力データによって表現することで、故障をシミュレートする工程を更に具えていることを特徴とする請求項１に記載のフォールトシミュレータに関する方法。
【請求項１７】　電気設計自動化システムと接続している相互接続されたＥＲＣＧＡを作動させる工程を更に具えていることを特徴としている請求項１に記載の方法。

【請求項１８】　前記相互接続されたＥＲＣＧＡを、メモリ回路に結合させる工程と、前記回路と接続している前記相互接続されたＥＲＣＧＡを動作させる工程とを更に具える請求項１に記載の方法。
【請求項１９】　前記双方向回路網を、双方向性相互接続を用いて、積の和に変換することによって、双方向性回路網を相互接続する工程を更に具えることを特徴とする請求項１に記載の方法。
【請求項２０】　ＥＲＣＧＡにおいて積を加算する工程を更に具えていることを特徴とする請求項１に記載の方法。
【発明の詳細な説明】
【０００１】
【発明の分野】本発明は、電気的に再構成可能なゲートアレイロジック素子（ＥＲＣＧＡ）の使用に関するとともに、複数のこのようなロジック素子の相互接続を具え、ラージデジタル回路網の電気的な表現を、シミュレーション、プロトタイピング、実行及び／又は演算のための相互接続されたロジック素子を用い、一時的に実際動作するハードウエアの構成に変換する方法にも関するものである。
【０００２】
【発明の背景及び概要】説明の便宜上、本出願では、リアライザシステム（Realizer System）として本発明を説明する。辞書には、後述するシステムの簡単な説明的名称が欠けている。リアライザシステムは、ハードウエアとソフトウエアとを具えており、シミュレーション、プロトタイピング、実行又は演算のために、ラージデジタルロジック回路網の表現を、一時的に実際動作するハードウエアの構成に変換する。（数個の最も広く利用することのできる構成可能なロジックデバイスを用いることによって、極めて多くのロジック機能を具えている場合、デジタルロジック回路網をラージとみなすこととしている。）（専らではなく）一般的に用いられている適切な用語を簡単に再検討することで、以下の説明を、より理解し易いものとすることができる。何かを“実現する”とは、それを実際又は現実のものとすることである。デジタルロジック回路網又は設計の全体又は一部を実現するということは、それを永久的に組み立てることなく、実際の動作を構成することである。“入力設計部”とは、実現されるべきデジタルロジック回路網を表している。この入力設計部は、計測デバイス又はユーザ指定実デバイスと同様に、組合せロジック及び記憶を表している基本要素と、基本要素の入出力ピン間の接続を表現している回路網とを具えている。ロジックチップ又は相互接続チップを“構成”するとは、その内部ロジック機能及び／又は相互接続を特定の方法で配置することである。入力設計部のためのリアライザシステムを構成するとは、その内部ロジック機能及び相互接続を、入力設計部に応じて配置することをいう。設計を“変換”するとは、その表現を構成データのファイルに変換することであり、これをリアライザハードウエアに直接用いると、設計部を実現することができる。設計部を“作動”させるとは、入力設計部の表現に応じて構成されたリアライザハードウエアを実際に作動させることである。
“相互接続”とは、ピンがワイヤで相互接続されているかのように、多数のチップＩ／Ｏピン間にロジック信号を通過させるための再構成可能な手段である。“パス”とは、部分的にクロスバー相互接続におけるロジックチップとクロスバーチップとの間の又は部分的クロスバーの階層におけるクロスバーチップ間の組込相互接続ワイヤの中の一つをいう。“パスナンバ”は、一対のチップを相互に接続している多くのパスの中から特定のパスを特定するものである。“ＥＲＣＧＡ”とは、電気的に再構成可能なゲートアレイ、すなわち組合せロジックの捕捉及び入力／出力接続（及び付加的な記憶装置）のことであり、その機能及び相互接続は、単に電気信号を供給することにより何回にも亘って構成及び再構成される。
“ロジックチップ”とは、リアライザシステムにおける入力設計部の組合せロジックと、記憶装置と、相互接続とを実現するのに用いられるＥＲＣＧＡである。“Ｌチップ”とは、ロジックチップ、又はロジックチップの場所に取り付けられるメモリモジュール、又はユーザ指定のデバイスモジュールである。“相互接続チップ”とは、Ｉ／Ｏピン間の任意の相互接続を実行することのできる電気的に再構成可能なデバイスである。“ルーティングチップ”とは、直接相互接続、又はチャンネルルーティング相互接続に用いられる相互接続チップのことである。“クロスバーチップ”とは、クロスバー相互接続又は部分的クロスバー相互接続に用いられる相互接続チップである。“Ｘチップ”とは、Ｌチップを相互に接続する部分的クロスバー中のクロスバーチップである。
“Ｙチップ”とは、階層部分的クロスバー相互接続の第２レベルにおけるクロスバーチップであり、Ｘチップを相互に接続している。“Ｚチップ”とは、階層部分的クロスバー相互接続の第３レベルにおけるクロスバーチップであり、Ｙチップを相互に接続している。“ロジックボード”とは、ロジックを伝達するプリント回路ボード及び相互接続チップである。“ボックス”とは、１以上のロジックボードを具えているカードケージのような物理的格納装置である。“ラック”とは、１以上のボックスを具えている物理的格納装置である。“システムレベル相互接続”とは、個々のチップより大きなデバイスを相互接続することであり、ロジックボード、ボックス、ラック等である。“ロジックセルアレイ”又は“ＬＣＡ”とは、ＥＲＣＧＡの特定の例であり、Xilink Inc.その他で製造され、好適な例に用いられるものである。
“構成可能なロジックブロック”又は“ＣＬＢ”は、構成可能なロジックの小さなブロック及びフリップフロップであり、ＬＣＡにおける組合せロジック及び記憶を表

している。“設計メモリ”とは、入力設計部において特定されるメモリ機能を実現するメモリデバイスである。“ベクトルメモリ”とは、多くの刺激信号をリアライザシステムに供給及び／又はリアライザシステムにおいて実現される設計部からの多くの応答信号を捕捉するのに用いられるメモリデバイスである。“スティミュレータ（stimulator）”とは、刺激信号を実現された設計部の個々の入力端子に供給するのに用いられるリアライザシステム中のデバイスである。“サンプラ”とは、実現された設計部の個々の出力端子からの応答信号を捕捉するのに用いられるリアライザシステム中のデバイスである。“ホストコンピュータ”とは、リアライザシステムのホストインターフェイスハードウエアが接続されている慣用のコンピュータシステムであって、リアライザのハードウエアの構成及び動作を制御するものである。“ＥＤＡシステム”とは、電気自動設計システム、すなわち電気設計部を作成、編集、及び分析するのに用いられるコンピュータベースのツールに関するシステムである。ホストＥＤＡシステムとは、リアライザシステムを応用する多くの場合において、入力設計ファイルを発生させるものである。

【０００３】シングルラージ設計部を保持するのに十分な容量を有する再構成可能なゲートアレイを用いれば、リアライザ技術の多くは不要である。しかしながら、２つの理由からこのことは決して実現することができない。まず第１に、ＥＲＣＧＡのロジック容量は、同じ製造技術を用いて製造された物理的に同一のサイズの再構成不可能な集積回路と同じではない。再構成のための機能は、チップのかなりのスペースをとる。ＥＲＣＧＡは、信号を導くスイッチングトランジスタと、これらのスイッチを制御するための記憶トランジスタとを有しており、ここで、再構成不可能なチップは金属トレースを具えている。そして、ＥＲＣＧＡは、これらのトランジスタをロジックとして用いることができる。再構成可能なチップに必要とされる規則性は、リソースが実際の設計では用いられていないということである。その理由は、規則的なロジック構造の配置及びルーティングが、利用可能なゲートを１００％用いることができないからである。ＥＲＣＧＡを製作するためのこれらの計数の結合は、再構成することのできないチップのロジック容量の約１／１０である。目下のところ、実際の実行では、ＥＲＣＧＡに要求される最大ゲート容量は９，０００ゲートである（Xilinx XC 3090）。同様の技術を用いて製造された、現在ほぼ慣用となっている集積回路では、１００，０００ゲートロジック容量以上が与えられる（モトローラ）。第２に、通常１０～１００或いはそれ以上の多くの集積回路を用いて、多くのプリント回路ボード上にリアルデジタルシステムを設けることはよく知られていることである。ＥＲＣＧＡのロジック容量が、大規模集積回路と同等である場合であっても、ほとんどのデジタルシステムを実現するのに、依然としてこのようなチップを多く用いることとなる。しかし、ＥＲＣＧＡのロジック容量は、大規模集積回路と同等ではないため、更に多くのチップが必要となる。最終的に、リアライザシステムがシングル大規模チップのロジック容量を有するためには、リアライザシステムが１０のオーダのＥＲＣＧＡを有している必要がある。このリアライザシステムが、このようなチップの容量を有しているためには、１００個のオーダのＥＲＣＧＡが必要とされる。このことは、特殊な製造技術とは無関係に必要とされることに注意しなければならない。チップ当たりのトランジスタの数を２倍にすることによる製造工程によって、ＥＲＣＧＡの能力を２倍にすると、再構成不可能なチップの容量は２倍となり、従って、設計サイズ全体も同様に２倍となる。

【０００４】これらの理由によって、有効なリアライザシステムを開発するためには、電気的に再構成可能な方法で、数百個のＥＲＣＧＡを相互接続できるようにするとともに、設計を数百個のＥＲＣＧＡの構成に変換できるようにする必要がある。本発明は、ＥＲＣＧＡそれ自体の技術にまで及ぶものではなく、多くのＥＲＣＧＡからリアライザシステムを開発するための技術にのみ関するものである。ＥＲＣＧＡ技術は、いかにしてリアライザシステムを開発するかを示してはいない。その理由は、それが別の問題だからである。一つのＩＣチップの全体を構成している再構成可能な相互接続ロジック素子のためのＥＲＣＧＡ技術は、多くを相互接続するのに適用できない。いずれか一方の方向に信号を通すスイッチングトランジスタによって、ＥＲＣＧＡ相互接続を容易に達成することができる。一つのチップ全体に亘って障壁が存在しないので、相互接続に利用する多数の通路が存在する。チップが小さいので信号のディレイも小さい。多くのＥＲＣＧＡを相互接続するのはむずかしい。その理由は、ＩＣパッケージピン及びプリント回路ボードを伴っているからである。利用することのできるピンの数が制限されているということは、相互接続のための通路の数が制限されているということである。チップの信号の入出力は、（例えば増幅しながら）アクティブピンバッファを介して行われなければならない。アクティブピンバッファは、一方向にのみ信号を送ることができる。これらのバッファ及び回路基板はトレースによって、一つのチップによって生ずるディレイよりも大きなディレイが生ずるリアライザシステムの相互接続技術によって、ＥＲＣＧＡとは全く別の方法で、これらの問題を解決する。最終的に、ＥＲＣＧＡ技術では、設計を多くのチップの構成に変換する必要はない。リアライザシステムの相互接続は、ＥＲＣＧＡ内の相互接続とは全く異なるものであり、相互接続を決定及び構成する全く別の方法が必要とされる。所定の時間に利用することのできる迅速且つ緻密なシリコン技術を用いて、ＥＲＣＧＡ

を開発する。（１ミクロンＳＲＡＭ技術を用いて、1989 Xilinx XC3000 ＬＣＡを開発する。）これは、実現される迅速且つ緻密なシステムと同一の技術である。ＥＲＣＧＡは汎用のものであり、再構成可能な相互接続を具えているので、現行のゲートアレイや慣用のチップほど緻密ではないファクタを具えている。リアライザシステムは、ＥＲＣＧＡのレベルより高い汎用性及び再構成可能性に対するサポートを反復する。従って、リアライザシステムは、現行の緻密なシステム程緻密ではなく、常に概略的には、１のオーダの一定のファクタとなっている。ボードレベルのリアライザシステムは、ゲートアレイを実現し、ボックスレベルのリアライザシステムは、ボード及び大規模慣用チップを実現する。更に、ラックレベルのリアライザシステムは、ボックスを実現する。設計構造は、パッケージングの影響を強く受ける。Ｉ／Ｏピン幅：ＶＬＳＩチップレベルでは、１００個のＩ／Ｏピンは容易に開発でき、２００ピンは開発が困難であるが、用いないこともなく、４００ピンに関しては、ほとんど開発されていない。ボードレベルでは、これらの数字は概して２倍となっている。ロジック密度：ボードが、しばしば５個のＶＬＳＩチップを具えており、１０個のＶＬＳＩを具えることも可能であるも、２０個のＶＬＳＩを具えることは一般的ではない。単にその理由は、実際のボードの最大値が約２００平方インチに制限されているからである。ボックスは、通常１０～２０ボードを具えており、時には４０ボードを具えている。相互接続密度：２次元ワイヤの平面を数枚用いることができる場合、モジュールは、完全にチップ及びボード上で相互接続される。しかし、背面が本質的に一次元的な場合には、ボックスレベルにおいて、それほど完全に相互接続されているわけではない。これらパッケージングの制約は、有用なリアライザシステムにおいて見られるシステム構造にかなり影響を及ぼす。リアライザシステムでは低密度であるがために、実現される設計部では、単一のロジックチップは、通常、唯一のモジュールを構成する。一つのボードにおけるロジックチップの複合体によって、１つ又は2つのＶＬＳＩチップを実現する。リアライザボードのボックスは、設計部において、単一のボードを実現する。更に、ボックスのラックは、設計部のボードのボックスを実現する。従って、リアライザシステムのボードレベルのロジック及び相互接続の複合体は、設計部のＶＬＳＩチップと同じロジック・相互接続容量並びにＩ／Ｏピン幅を有している必要がある。リアライザシステムのボックスは、設計部のボードと同じロジック・相互接続容量及びＩ／Ｏピン幅を必要とし、リアライザシステムのラックは、設計部のボックスと同じロジック・相互接続容量を必要としている。

【０００５】

【発明の実施の形態】内容一覧表

【０００６】１．リアライザハードウエアシステム

リアライザハードウエアシステム（図１）は：

１）１）少なくとも二つのロジックチップ（通常、数十個又は数百個）及び

２）付加的に、メモリモジュール、ユーザ指定のデバイスモジュールのような、１以上の特定目的のための構成要素を具えている１セットのＬチップを、

２）Ｉ／Ｏピンを相互接続可能なすべてのＬチップに接続されている構成可能な相互接続と、

３）ホストコンピュータ、構成システム及びデータ入出

力又は制御のためのホストが使用することのできるすべてのデバイスに接続されたホストインタフェースと、4）ホストインタフェース、すべての構成可能なLチップ及び相互接続デバイスに接続された構成システムとを具えている。このハードウエアを、通常、ロジックボード、ボックス及びラックの形態で実装し、ホストコンピュータに接続する。このハードウエアは、ホストコンピュータの制御の下で作動する。

【0007】1.1 ロジック及び相互接続チップ技術

1.1.1 ロジックチップデバイス

デバイスが、リアライザロジックチップとして役立つためには、このデバイスが電気的に再構成可能なゲートアレイ（ERCGA）でなければならない：

1）デバイスは、容量制限を条件として、組合せロジック（及び付加的な記憶装置）を具えているデジタルロジック回路によって構成することができなければならない。

2）デバイスは、その機能及び内部相互接続を、電気的に何回でも、種々の論理回路に適合するように構成することができるという点において、電気的に再構成可能でなければならない。

3）デバイスは、特定の回路網又は特定するI／Oピンとは無関係に、デジタル回路網でI／Oピンを自由に接続し、リアライザシステムの部分クロスバー、又は直接相互接続が首尾よくロジックチップを相互接続できなければならない。

ロジックチップとして好適な、再構成可能なロジックチップの一例としては、ロジックセルアレイ（Logic Cell Array（LCA））がある（"The Programmable Gate Array Handbook",Xilinx Inc.,San Jose,CA,1989)。このロジックチップは、Xilinx Inc.その他で製造されている。このチップは、構成可能なロジックブロック（Cnofigurable Logic Block（CLB））の2次元配列を具えている。この2次元配列は、再構成可能なI／Oブロック（IOB）で囲まれているとともに、CLBとIOBとの間の行と列とに配置されたセグメントを配線することによって相互接続されている。各CLBは、数個の入力端子と、ロジック機能を再構成することのできる多重入力組合せロジック回路網と、1以上のフリップフロップと、CLB内の再構成可能な相互接続によって連結することができる1以上の出力端子とを具えている。各IOBを再構成し、チップの入力バッファ又は出力バッファとすることができる。更に、各IOBを外部I／Oピンに接続する。配線したセグメントをCLB、IOB及び、お互いに接続することによって、再構成可能なパストランジスタ及び相互接続マトリックスを介し、CLB、IOB及びセグメント間に、相互接続を形成する。すべての再構成可能な機能を、チップのシリアルシフトレジスタにおけるビットで制御する。従って、LCAは"構成ビットパターン"のシフトによって完全に構成される。構成に要する時間は10～100マイクロ秒である。Xilinx 2000及び3000シリーズのLCAは、64～320のCLBを具えており、56～144のIOBを使用することができる。LCAネットリスト（netlist）変換ツール（以下にて示す）は、ロジックをCLBに作成し、CLBとIOBとの間の相互接続を最適にしている。CLBとI／Oピンとの間の相互接続を構成することによって、LCAは、特定の回路網又は特定するI／Oピンとは無関係に、デジタル回路網でI／Oピンを自由に接続することができる。リアライザシステムを好適に具体化するには、LCAデバイスをそのロジックチップとして用いる。ロジックチップとして好適な他の種類のアレイとしては、ERA、すなわち電気的に再構成可能なアレイがある。市販されているものとしてはplesseyのERA60Kのタイプのデバイスがある。これは、構成ビットパターンを部分的にRAMにロードすることで構成される。ERAを2入力NANDゲートのアレイとして構成する。RAMの値に応じて、2入力NANDゲートの各々を独立に互いに相互接続する。ERAは、ゲート入力端子の一連の相互接続通路への接続を切換える。ERA 60100は、約10,000個のNANDゲートを具えている。アレイの周辺のI／Oセルは、ゲート入力端子及び／又は出力端子を外部I／Oピンに接続するのに用いられる。ERAネットリスト変換ツールは、ロジックをゲートに作成し、ゲート間の相互接続を最適にするとともに、以下に示すように、構成ビットパターンファイルを出力する。ゲートとI／Oセル間の相互接続を構成可能にすることによって、ERAは、特定の回路網又は特定するI／Oピンとは無関係にデジタル回路網を用いてI／Oピンを自由に接続することができる。ロジックチップとして用いることのできる、更に他の種類の再構成可能なロジックチップとしては、EEPLD、すなわち、電気的に消去可能で、プログラム可能なロジックデバイス（"GAL Handbook", Lattice Semiconductor Corp., Portland, OR,1986）がある。商用のものとしては、ラティス・ジェネリック・アレイ・ロジック（Lattice Generic Array Logic（GAL））がある。これは、ビットパターンを、ロジック構成の部分にロードすることで構成される。GALは、出力フリップフロップを有する、積の和のアレイとして構成されており、その構成は、Xilinx LCAよりも汎用性を有していない。GALによって、I／Oピンを、すべての入力ピン間及びすべての出力ピン間のロジックに接続せずにすみ、部分的に要件を満足している。GALは、10～20個のピンを有しており、比較的小さな構造となっている。しかし、GALはリアライザロジックチップとして用いられる。プログラム可能なロジックチップについての詳細は、米国特許第4,642,487号、第4,700,187号、第4,796,216号、第4,722,084号、第4,724,307号、第4,758,

９８５号、第4,768,196号及び第4,786,904号明細書において説明されている。ここでは、これらの明細書の内容を説明に用いている。

【０００８】１．１．２　相互接続チップデバイス

相互接続チップは、クロスバー相互接続の全体及び一部に用いるクロスバーチップと、直接相互接続及びチャンネルルーティング相互接続に用いるルーティングチップとを具えている。デバイスが、リアライザ相互接続チップとして役立つためには：

１）デバイスは、直ちに、任意に選択されたＩ／Ｏピンのグループ間で多くのロジック相互接続を形成し、各相互接続は、その入力Ｉ／Ｏピンからロジック信号を受信するとともに、これらの信号を出力Ｉ／Ｏピンに供給できなければならない。

２）デバイスは、その相互接続を電気的に規定するという点において、再構成可能でなければならず、多くの種々の設計に適合できるように再規定できなければならない。

３）クロスバー加算技術を用いて、部分的クロスバー相互接続におけるトライステート回路網を相互接続する場合、デバイスは、加算ゲートを具体化できなければならない。（クロスバー加算技術を用いない場合には、トライステート・セクションにて説明するように、他のトライステート技術を用いる。）

上述したＥＲＣＧＡデバイス、すなわち、ＬＣＡ、ＥＲＡ及びＥＥＰＬＤは、これらの要件を満足しており、相互接続チップとして用いられる。相互接続チップにロジックをほとんど或いは全く用いない場合、ほとんどのデジタル回路網を構成することのできる機能は、データを直接入力ピンから出力ピンへと送ることができる。ＬＣＡは、リアライザシステムを好適に具体化する際、クロスバーチップとして用いられる。TI 74AS 8840デジタルクロスバースイッチ（SN 74 AS 8840 Data Sheet,Texas Instruments, Dallas, TX, 1987）、すなわち、通常電話スイッチに用いられる交差点スイッチデバイスを、相互接続チップとして用いることができる。ところで、これらのクロスバースイッチデバイスを、作動中、動的に変化する構成に応用する場合、データ伝送スピードに匹敵する再構成スピードが得られる。この再構成スピードは、ＥＲＣＧＡデバイスの構成スピードよりも速い。この結果、このようなクロスバースイッチデバイスは、ＥＲＣＧＡよりも高価且つ低容量であり、あまり望ましくないリアライザ相互接続チップを作成することとなってしまう。

【０００９】１.１.３　ＥＲＣＧＡ構成ソフトウエア

構成ビットパターンは、ユーザ指定に従ってＥＲＣＧＡにロードされ、そのロジックを構成する。ユーザが単独でロジックを構成するのは非現実的である。従って通常ＥＲＣＧＡ装置を製造することによって、ネットリスト変換ソフトウエアツールが得られる。このツールは、ネットリストファイルに具わっているロジック仕様を、構成ビットパターンファイルに変換する。リアライザ設計変換システムは、ＥＲＣＧＡのコンピュータメーカによって提供されるネットリスト変換ツールを用いている。リアライザ設計変換システムが、設計部において、ネットリスト変換ツールを読出して変換し、ロジックチップに分割し、さらに相互接続を決定すると、各ロジックに対するネットリスト及びリアライザハードウエアにおける相互接続チップを発生させる。ネットリストファイルとは、すべての基本要素（ゲート・フリップフロップ及びＩ／Ｏバッファ）及び、単一ロジックチップ又は相互接続チップで構成されるこれらの相互接続のリストのことである。リアライザ設計変換システムは、ＥＲＣＧＡネットリスト変換ツールを各ネットリストファイルに供給し、各チップの構成ファイルを得る。ロジックチップ及び相互接続チップとして種々のデバイスを用いる場合には、適切なツールを用いる。構成ファイルは、２進ビットパターンを具え、これはＥＲＣＧＡデバイスにロードされると、ネットリストファイルの仕様に応じてファイルを構成する。ＥＲＣＧＡデバイスは、これらのファイルを、永久に記憶され且つ作動前に設計部のリアライザシステムを構成するのに用いられる単一バイナリーファイルに収集する。リアライザ設計変換システムは、ツールのＥＲＣＧＡコンピュータメーカによって規定されるネットリスト及び構成ファイルフォーマットに準拠している。

【００１０】１.１.４　ネットリスト変換ツール

リアライザシステムを好適に具現化するために、ロジックチップ及びクロスバーチップとしてＬＣＡを用いているので、Xilinx　ＬＣＡネットリスト変換ツール及びそのファイルフォーマットをここで説明する。Xilinx製のＬＣＡネットリスト変換ツール（ＸＡＣＴ）によって、ネットリスト形式のロジック回路網が与えられるとともに、自動的にロジック素子がＣＬＢに作成される。Ｉ／Ｏピンの位置に関し、最適の方法でロジック素子を構成し、内部相互接続を容易にすることができる。従って、このツールは、いかにしてロジックチップの内部相互接続を構成するかを解明し、その出力結果としての構成ファイルを作成する。ＬＣＡネットリスト変換ツールは、単に個々のＬＣＡを変換するだけであって、ロジック回路網が大き過ぎて単一のＬＣＡに適合できない場合には障害が生じる。Xilinx　ＬＣＡネットリストファイルをＸＮＦファイルと称する。これはアスキーテキストファイルであり、各々の基本要素に対する１セットのＸＮＦファイル中のステートメントを具え、基本要素、ピン及びこれらのピンに接続される回路網の名称を特定する。これらの回路網は、ＬＣＡネットリスト中で相互接続されており、入力設計部の回路網ではなく、ＬＣＡ基本要素を接続している。ＸＮＦファイル中のいくつかのファイルは、設計変換の結果、入力設計部の回路網に直接対

応しているが、他のファイルは対応していない。例えば、これらは‘I_1781‘と称する、2入力XORゲートを特定するためのXNFファイル基本要素ステートメントであり、前記2入力XORゲートの入力ピンを、‘DATA0’及び‘INVERT’と称する回路網に接続し、その出力ピンを、‘RESULT’と称する回路網に接続している：

SYM,I_1781,XOR
PIN,O,O,RESULT
PIN,1,I,DATA0
PIN,0,I,INVERT
END

入力及び出力I／OピンバッファI（入力のためのIBUF及び出力のためのOBUF）は、I／Oピンを特定するためのステートメントを付加することで、同様にして特定される。これらは、OBUFに対する基本的なステートメントであり、これによって、‘RESULT’を‘RESULT_D’と称する回路網を介してI／Oピン‘P57’において駆動させる：

SYM,IA_1266,OBUF
PIN,O,O,RESULT_D
PIN,I,I,RESULT
END
EXT,RESULT_D,O,LOC=P57

Xilinx LCAを、RBTファイルと称する。これは、アスキーテキストファイルであり、構成される部分を識別するヘッダステートメントと、動作のための部分を構成するのに用いられるバイナリービットパターンを特定する‘0’及び‘1’のストリームとを具えている。

【0011】1.2 相互接続アーキテクチャ

実際の場合、大規模入力設計部を実現するためには、多くのロジックチップを使用しなければならないため、リアライザのロジックチップを再構成可能な相互接続に接続しなければならない。この相互接続によって、必要とされているように、設計部中の信号が、分離ロジックチップ間を流れる。この相互接続は、電気的相互接続及び／又は相互接続チップの結合を具えている。リアライザシステムにおいて大規模設計部を実現するためには、トータルで幾万ものI／Oピンを有するロジックチップが相互接続によって供給されなければならない。相互接続は、システムサイズが大きくなるにつれて経済的に拡張可能であり且つ容易なものであるとともに、入力設計部を幅広く確実に構成することができ、更に高速であり、ロジックチップ間の遅延を最小にすることができる。現実の設計部における回路網単位のピンの平均数は、設計部のサイズとは無関係な小さな数であるため、接続するロジックチップのトータル数が増加するにつれて、優れた相互接続のサイズ及びコストも直接的に増加するはずである。設計部の容量が増大するにつれて、用いる特定ロジックチップ容量、ロジックチップの数及びロジックチップピンの数も直接的に増大する。従って、設計部の容量とともに、優れた相互接続のサイズ及びコストも直接的に変化する。2クラスの相互接続構造を説明する：隣接相互接続を第1セクションで説明し、クロスバー相互接続を次のセクションで説明する。最も近い隣接相互接続は、ロジックチップと、2次元、3次元又はそれ以上の次元の面に従って、混合及び構成された相互接続とによって構成される。最も近い隣接相互接続では、ゲートアレイチップの行列編成又はプリント回路基板をロジックチップの編成にまで拡張している。所定の入力設計部の構成は、チップ及びボードを開発する場合に用いられるのと同様の配置及びルーティングプロセスによって決定される。クロスバー相互接続は、相互接続されているロジックチップとは異なる。クロスバー相互接続は、伝送及び演算に用いられるクロスバーの多入力多出力編成に基づくものであり、平面的に構成することができる。最も近い隣接相互接続は、ロジック容量が大きくなるにつれて大きくなるが、ルーティング通路が密集するにつれて大規模相互接続はゆっくりとなり、また、構成を決定することが困難且つ不確実なものとなる。単なるクロスバーは、その直接性のために極めて高速であり、その規則性のために構成が容易であるが、すぐに非実用的な大きさとなってしまう。部分的なクロスバー相互接続は、ほとんどの直接性と、単なるクロスバーの規則性とを保持するが、設計部容量の増大とともにのみ直接的に増大し、理想リアライザ相互接続を実現している。実際のリアライザシステムでは、図示した以外の相互接続を用いることはできるが、部分的クロスバーを好適な具体例に用いる。この使用は、この明細書全体を通して類推される。

【0012】1.2.1 最も近い隣接相互接続

1.2.1.1 直接相互接続

直接相互接続では、相互接続チップを用いずして、直接すべてのロジックチップを、規則的なアレイにおいて相互に接続する。この相互接続は、単にロジックチップ間の電気的接続から成っている。ロジックチップの相互接続は、多くの異なるパターンを形成することができる。一般的に、一つのロジックチップのピンをグループ毎に分割する。従って、すべてのロジックチップにおいて各ピンのグループを、他のロジックチップの同様なピンのグループ等に接続する。各ロジックチップは、単に一組のすべてのロジックチップ、すなわち、物理的な意味において、つまり、少なくともアレイの接続形態という意味において、最も近い隣接ロジックチップとだけ接続する。1以上のロジックチップにロジックを接続するすべての入力設計部回路網を、相互接続チップとしての機能を果たす他のロジックチップに、これらのロジックチップを直接接続する場合には直接接続し、又は、一連の他のロジックチップを介して接続し、チップの実現するロジックのいずれかに接続することなしに、ロジック信号

を一方のＩ／Ｏピンから他方のＩ／Ｏピンへ伝達する。このようにして、いかなる所定のロジックチップも、設計部ロジックの共有に加えて、一方のチップから他方のチップへ相互接続信号を伝達するように構成されている。相互接続機能を実行することのできない非ロジックチップリソースを、アレイの周辺で、専用ロジックチップピンに接続又は、ロジックピンを相互に接続しているピンに正接させ接続している。図２に示す特定の例では、行及び列の２次元格子中に配置されたロジックチップを具えており、その各々のチップは、メモリを有する隣接ロジックチップに北側、南側、東側及び西側で接続されている４つのピンのグループと、Ｉ／Ｏと、周辺で接続されたユーザ指定のデバイスとを具えている。この相互接続を、ここで説明した２次元のものから、より高次元のものへと拡張することができる。一般的に、'ｎ'を次元の数とする場合、各々のロジックチップのピンを、２＊ｎ個のグループに分割する。各々のロジックチップは、規則的な形態で、２＊ｎ個の他のロジックチップと接続している。他の変更も同様であるが、ピンのグループの大きさは等しくない。ロジックチップの数及び各々のピンの数に基づき、ピングループサイズの寸法及びセットを選択し、２のロジックチップ間に介在するロジックチップの数を最小にするとともに、各々直接隣接しているチップ対の間を充分に相互接続し、回路網がこれら二つのチップだけにつながるようにしている。相互接続のためのロジックチップをいかに構成するかを決定するには、ロジックのためのチップをいかに構成するかを決定しなければならない。ロジックチップを構成するためには：

１）基本要素変換のセクションで述べたように、設計部のロジックをロジックチップの基本要素形態に変換する。

２）ロジックチップにおけるロジック基本要素を、区分化及び配置する。ロジックチップのロジック容量内に各々適合しているサブ回路網に、設計部を区分化することに加えて、サブ回路網をお互いに対して配置し、必要とされる相互接続の量を最小とする。ゲートアレイ又は標準セルチップ自動区分化及び配置ツール（"Gate Station Reference Manual"，Mentor Graphics Corp,1987）において用いられているような、標準区分化及び配置ツール方法を用いて、いかにしてロジック基本要素をロジックチップに割り当てるかを決定し、相互接続を達成する。このことは、定評のある方法であり、ここでは、これ以上の説明を省略する。

３）ロジックチップ間の相互接続を配線する。すなわちロジックチップの中から特定のロジックチップ及びＩ／Ｏピン相互接続を選定し、ゲートアレイ又は標準セルチップ自動ルーティングツール（"Gate Station Reference" Manual",Mentor Graphics Corp., 1987 ）のような標準ルーティングツールを用い、いかにしてチップを構成するかを決定し、相互接続を達成する。これは定評のある方法であるため、いかにしてこの方法を相互接続の問題に適用するかを除き、ここではこれ以上の説明を省略する。ロジックチップのアレイを、シングルラージゲートアレイ又は標準セルチップと同じ方法で取り扱う。各々区分化されたロジックサブネットワークは、大規模ゲートアレイロジックマクロに対応しており、相互接続されたロジックチップＩ／Ｏピンは、ルーティングに用いる配線チャンネルを規定している。特に、各々のルーティング方向には、相互接続されたロジックチップＩ／Ｏピンの各グループ毎のピンと同数のチャンネルを具えている。ロジックチップ間では多くの相互接続が可能であるので、多くのルーティング層によって、ゲートアレイのチャンネル制約を取り除くのと同様の方法を用いて、ルーティングを制約することなく、各末端部において同じチャンネルを用いる。

４）ルーティングの過剰（ルーティング処理の間或るポイントにおいてチャンネルをルーティングすることができない場合）のために相互接続を行うことが不可能な場合、調整された基準を用いて、設計部を再区分化及び／又は再配置し、過剰を除去し、再び相互接続を試みる。

５）どの回路網がどのチャンネルを使用するかについての仕様を、特定のルーティングチャンネルとＩ／Ｏピンとの間の対応に応じて、個々のロジックチップに対するネットリスト及び、ロジックチップ信号に対する特定ピンの役割に変換する。ロジック基本要素の仕様とともに、Ｉ／Ｏピン仕様及びロジックチップ内部相互接続の形態の仕様を、各ロジックチップ毎のネットリストファイルに送出する。

６）ロジックチップネットリスト変換ツールを用い、各ロジックチップ毎の構成ファイルを発生させるとともに、これらを組合せ、入力設計のための最終的なリアライザ構成ファイルを作成する。

【００１３】１.２.１.２　チャンネルルーティング相互接続

チャンネルルーティング相互接続は、直接相互接続の変形である。この場合、チップは、ロジックとしては用いられず単に相互接続を行う相互接続チップと、専らロジックとして用いられるロジックチップとに分割される。特に、ロジックチップは、お互いを直接接続するのではなく、その代わりに、ただ単に相互接続チップを接続するだけである。その他すべての点において、チャンネルルーティング相互接続は、直接相互接続方法に従って作成されている。１以上のロジックチップ及び回路網は、ルーティングチップと称する一連の相互接続チップを構成することによって相互接続する。ルーティングチップは、これらのロジックチップを接続させるとともに、お互いを接続させ、ロジックチップＩ／Ｏピン間にロジック的接続が確立される。このことは、構成可能な'回路基板'に用いられる。チャンネルルーティング相互接続

を、一例として２次元としている：すなわち、図３に示されているように、ロジックチップは、行及び列の形態で配置されており、その周囲はルーティングチップによって完全に囲まれている。アレイを、全てがルーティングチップで構成されている行と、ロジックチップ及びルーティングチップで交互に構成されている行とで交互に構成する。このようにして、ロジックチップの周囲には、行方向及び列方向に、切目なくルーティングチップが配置されている。各チップのピンを４つのグループ、すなわち、“北側、東側、南側、西側”と称する４つのエッジに分割する。各々のチップのピンを、４つの最も近い隣接チップに格子状に接続する。：すなわち、北側のピンを北側に隣接するチップの南側ピンと接続し、東側ピンを東側に隣接するチップの西側ピンに接続する。以下同様である。このモデルは、上記例の２次元より大きな次元にまで拡張することができる。一般的には、‘ｎ’を次元の数とする場合、各ロジックチップのピンは、２＊ｎ個のグループに分割される。各ロジックチップは２＊ｎ個の隣接チップに接続している。アレイの中心においては各々のロジックチップに対して（２＊＊ｎ－１）個のルーティングチップが存在する。ロジックチップとルーティングチップとの特徴に基づき、このチャンネルルーティングモデルの一般化を同様にして用いる。ロジックチップのピンを数個のグループに分けることができる。ルーティングチップのピンも数個のグループに分けることができる。但し、ルーティングチップのピンのグループ数が、ロジックチップのピンの数と同じである必要はない。ロジックチップとルーティングチップとは、同数のピンを具えている必要はない。ロジックチップとルーティングチップとの規則的なアレイであり、且ついかなる所定のロジックチップであっても、それが最も近い隣接チップの限定セットとだけ接続されている限り、これらの変形が適用される。ロジックチップ間の相互接続を、ロジックチップを介してではなく、相互接続チップを介してのみ配線するということを除いて、直接相互接続に用いると同様の方法を用い、ロジックチップをいかに構成するかを決定するとともに、相互接続チップをいかに構成するかを決定する。回路網のロジック信号は、相互接続を完成させるのに必要なルーティングチップと同数のルーティングチップを介して流れる。各々のルーティングチップによって信号の伝搬が遅れるので、信号が流れるルーティングチップが増えれば増える程、相互接続を介しての信号伝搬遅れ時間は長くなる。ルーティングの必要条件を最小とできるように、ロジック設計部を区分化するとともに、それぞれの区分を特定のロジックチップに配置するのが一般的には望ましい。ルーティングが過剰であり、相互接続を行うことができない場合、調整された基準を用いて設計部を再区分化及び／又は再配置し、再び相互接続を行う。このサイクルは、必要な限り繰り返される。

【００１４】１.２.２　クロスバー相互接続

１.２.２.１　完全クロスバー相互接続

クロスバーとは、制約なくピンを他のピンと接続することのできる相互接続アークテクチャーである。これは、コンピュータ及び通信デバイスのスイッチング回路網において、メッセージを通信するために広く用いられている。完全なクロスバーとして構成される相互接続は、すべてのロジックチップピンに接続するとともに、いかなるピンの相互接続の組合せであっても構成可能な相互接続によって、いかなる入力設計及びロジックチップ区分化であっても、直接的に相互接続を達成することができる。その理由は、いかなるピンであっても、いかなる他のピンに直接接続することができるからである。しかし、多くのロジックチップを相互接続することのできる実用的な単一デバイスは存在しない。例えば、好適例のロジックボードは、各々接続すべき１２８個のピンを有するロジックチップを１４個具えている。合計で１７９２ピンとなり、実用的なシングルチップの容量をはるかに超えている。実用的な相互接続チップ及びデバイスからクロスバーを構成することができる。これらを構成することによって、Ｉ／Ｏピン間に任意の相互接続を実現することができる。クロスバー相互接続の場合、これらをクロスバーチップと称する。実用的なクロスバーチップからクロスバー相互接続を構成する一般的な方法は、一つのクロスバーチップを用いて、一つのロジックチップピンをクロスバーチップが有するピンと同数の他のロジックチップピンと相互接続する。図４は、わかり易くするために極めて簡略化した一例を示している。各々８個のピンを有する４個のロジックチップを相互接続する。各々９個のピンを有するクロスバーチップを用いる。３個のクロスバーチップの最も左側の列によって、ロジックチップ４のピンＨをロジックチップ１、２及び３のピンと接続する。次の列によって、ピンＧ等をロジックチップ４のピンＧに接続する。同じロジックチップに関しては、内部で接続できることから、ピンと他のピンとを接続する必要はない。クロスバーチップの隣接する８個の列は、ロジックチップ３とロジックチップ１及び２とを相互接続している。ロジックチップ４は含まれていない。その理由は、ロジックチップ４のピンを、クロスバーチップの最初の８個の列によって、ロジックチップ３のピンに接続しているからである。最後の８個の列はロジックチップ１と２とを相互接続している。合計４８個のクロスバーチップを用いる。入力設計に基づく二つの回路網は、相互接続された状態を示している。回路網Ａは、ロジックチップ１のピンＤによって駆動され、ロジックチップ４のピンＢによって受信される。１で示されているクロスバーチップは、これらのピンの両方を接続しており、ロジックチップ１のピンＤから受信し、受信したものを、チップ４のピンＢに伝達する。このようにして、ロジック接続を構成する。回路網Ｂは、

ロジックチップ２のピンＦによって駆動され、ロジックチップ３のピンＧ及びロジックチップ４のピンＧによって受信される。クロスバーチップ２は第１相互接続を行い、クロスバーチップ３は第２相互接続を行う。一般的に、必要とされるクロスバーチップの数を予測することができる。各々Ｐ１個のピンを有するＬ個のロジックチップが存在し、且つ、１個のロジックチップピンをできる限り多くの他のロジックチップピンと各々接続できるようにしているクロスバーチップがＰｘ個のピンを具えている場合：

１）ロジックチップ１の中の一つのピンを、２からＬまでのロジックチップの（Ｌ－１）Ｐ１個のピンに接続しなければならない。これには、（Ｌ－１）Ｐ１／（Ｐｘ－１）個のクロスバーチップが必要とされる。すべてのピンを接続するには、（Ｌ－１）Ｐ１²／（Ｐｘ－１）個のクロスバーチップを必要とする。

２）ロジックチップ２の各々のピンを、３からＬまでのロジックチップの（Ｌ－２）Ｐ１個のピンに接続しなければならない。これには、（Ｌ－２）Ｐ１²／（Ｐｘ－１）個のクロスバーチップを必要とする。

３）ロジックチップＬ－１の各々のピンを、ロジックチップＬのＰ１個のピンに接続しなければならない。これには、Ｐ１²／（Ｐｘ－１）個のクロスバーチップを必要とする。

４）$X=(L-1)P1^2/(Px-1)+(L-2)P1^2/(Px-1)+\cdots$

$\cdots+P1^2/(Px-1)$

$=(L2-L)P1^2/2(Px-1)$

クロスバーチップの数Ｘは、ロジックチップの数の二乗とロジックチップ毎のピンの数の二乗とをかけ算したものが増加するにつれて、増加する。好適実施例のロジックボード（各々１２８個のピンを有する１４個のロジックチップ）は、各々１２９個のピンを有する１１６４８個のクロスバーチップ又は各々６５個のピンを有する２３２９６個のクロスバーチップを必要としている。クロスバー相互接続は、有用なリアライザシステムに用いるには、大規模且つ高価なものであり、非実用的である。

【００１５】１.２.２.２　完全クロスバー回路網相互接続

相互接続すべき設計回路網の数がロジックチップピンの合計数の１／２を決して超えることができないということを認識することによって、クロスバー相互接続の大きさを小さくすることができる。クロスバー回路網相互接続は、ロジック的には２つのクロスバーによって構成されており、その各々は、すべてのロジックチップピンを相互接続回路網（ＩＣＮ）と称する１セットの接続回路網に接続しており、ロジックチップピンの総数の１／２に番号を付している。１セットのロジックチップピンを、１セットのＩＣＮに接続するクロスバーチップが１セットのＩＣＮから、これらのピンへ接続を戻すこともできる（相互接続チップの一般性の撤回）ため、この相互接続をクロスバーチップで構成することもできる。各々のクロスバーチップは、１セットのロジックチップピンを１セットのＩＣＮと接続している。図５は、図４にて示したものと同一の４個のロジックチップを相互接続した一例を示す図である。各々８個のピンを有するクロスバーチップを用い、１６個のＩＣＮを設ける。３２個のクロスバーチップの各々は、４個のＩＣＮを用いて４個のロジックチップピンを接続する。回路網Ａを、クロスバーチップ１によって相互接続し、ロジックチップ１のピンＤから受信し、受信したものをＩＣＮに伝達するように構成する。また、回路網Ａを、クロスバーチップ２によって相互接続し、前記ＩＣＮから受信し、ロジックチップ４のピンＢを駆動する。このようにしてロジック接続を確立する。回路網Ｂは、ロジックチップ２のピンＦによって駆動され、クロスバーチップ４を介してロジックチップ３のピンＧで受信されるとともに、クロスバーチップ５を介してロジックチップ４のピンＧで受信される。好適実施例のロジックボード（各々１２８個のピンを有する１４個のロジックチップ）のクロスバー回路網相互接続は、各々１２８個のピンを有する３９２個のクロスバーチップ、又は各々６４個のピンを有する１５６８個のクロスバーチップを必要とする。クロスバー回路網相互接続では、使用するクロスバーチップの数は単なるクロスバーよりも少ない。クロスバー回路網相互接続の大きさは、ロジックチップの数と、ロジックチップピンの総数との積が増加するにつれて、大きくなる。これは、ロジックチップ数の二乗に達する。これは、純粋なクロスバーよりは優れているも、依然として望まれる直接スケーリングではない。

【００１６】１.２.２.３　部分的クロスバー相互接続

ロジックチップそれ自体によって、クロスバーが開発できない付加的な自由度を提供することができる。その理由は、ロジック回路網の所定の入力又は出力を、いかなるＩ／Ｏピンをも用いることができるように構成することができるからである。すなわち、特定の回路網とは無関係に構成するからである。この自由度によって部分的クロスバー相互接続をすることができる。これが、自由度をロジックチップの定義中に明示している理由である。部分的クロスバー相互接続では、各ロジックチップを分割するのと同様にして、各ロジックチップのＩ／Ｏピンを適切なサブセットに分割する。各クロスバーチップのピンを、各ロジックチップの各々から同じピンのサブセットに接続する。このようにして、クロスバーチップ‘ｎ’を、各ロジックチップのピンのサブセット‘ｎ’に接続する。サブセットと同数のクロスバーチップを用いる。各々のクロスバーチップは、サブセットのピンの数とロジックチップの数とをかけ算した数と同数

のピンを有している。各ロジックチップ／クロスバーチップ対を、各サブセット中のピンと同数のパスと称するワイヤで相互接続する。各クロスバーチップを各ロジックチップのピンと同一のサブセットに接続しているため、一つのロジックチップのピンにおける一つのサブセット中のＩ／Ｏピンから、もう一つのロジックチップのピンにおける別のサブセット中のＩ／Ｏピンへの相互接続を構成することはできない。このことは、相互接続すべき各々のロジックチップのピンの同一のサブセットから、Ｉ／Ｏピンを用い、各々の回路網を相互接続し、適宜にロジックチップを構成することによって避けられる。回路網に接続されるロジックチップ中に構成されるロジックチップに割り当てることのできるいかなるＩ／Ｏピンを用いても、ロジックチップを構成できるようにするため、一方のＩ／Ｏピンは、他方のＩ／Ｏピンと同様のものである。一般的なパターンを図６に示す。この図において、ロジックチップとクロスバーチップとを接続している各々のラインは、ロジックチップピンのサブセットを示している。各クロスバーチップをすべてのロジックチップのピンのサブセットに接続する。逆に言えば、このことは、各ロジックチップをすべてのクロスバーチップのピンのサブセットに接続していることを示している。これらの例では、クロスバーチップの数がロジックチップの数と等しい必要はない。好適な実現例では、このようなことは言えない。図７は、図１及び図２と同一の４個のロジックチップを相互接続している例を示している。各々８個のピンを有する４個のクロスバーチップを用いる。各クロスバーチップは、各ロジックチップにおいて同一の２個のピンを接続している。クロスバーチップ１をロジックチップ１～４の各々のピンＡ及びＢに接続する。クロスバーチップ２をすべてのピンＣ及びＤに接続し、クロスバーチップ３をすべてのピンＥ及びＦに接続するとともに、クロスバーチップ４をすべてのピンＧ及びＨに接続する。前記例の設計回路網Ａでは、ロジックチップ４のピンＢにおいて受信が行われるが、回路網Ａをロジックチップ１のピンＤにおけるドライバに相互接続することのできるクロスバーチップは設けられていない。いかなるＩ／Ｏピンであっても、回路網Ａを受信するロジックチップ４において構成されるロジックに割り当てることができるので、ピンＣはピンＢと同様であり、これを、他の回路網に用いることができる。結果的に、回路網Ａは代わりにピンＣによって受信され、クロスバーチップ２を構成することで、相互接続を達成する。設計回路網Ｂは、ロジックチップ３のピンＧ及びロジックチップ４のピンＧによって受信されるが、この回路網Ｂを、ロジックチップ２のピンＦにおけるドライバと相互接続できるクロスバーチップは設けていない。回路網Ｂは、代わりにピンＨによって駆動され、クロスバーチップ４を構成することで相互接続を達成する。好適な実施例では、部分的クロスバー相互接続を用いている。このロジックボードは、各々１２８個のピンを有する１４個のロジックチップを具えており、各々５６個のピンを有する３２個のクロスバーチップによって相互接続されている。ロジックチップピンを、各々４個のピンを有する３２個の適切なサブセットに分割するとともに、各クロスバーチップのピンを、各々４個のピンからなる１４個のサブセットに分割する。クロスバーチップ‘ｎ’を各ロジックチップピンのサブセット‘ｎ’に接続し、各ロジックチップ／クロスバーチップ対を４個のパスによって相互接続する。すべてのクロスバー相互接続の中で、部分的クロスバーの使用するクロスバーチップの数は最小である。部分的クロスバーのサイズは、ロジックチップピンの総数が増大するに従って直接的に増大する。このことは、ロジックチップの数、更にはロジック容量に直接関連するものであり、望ましい結果である。これを使用するのは比較的容易なことである。その理由は、部分的クロスバーが規則的であり、そのパスをテーブルで表現することが可能であり、更に特定の相互接続をいかにして決定するかは、単にパスの最適なペアをテーブルで捜すだけだからである。

【００１７】１.２.２.４　部分的相互接続の機能

部分的なクロスバー相互接続は、完全クロスバーが処理できるのと同数の回路網を処理することはできない。ソースロジックチップにおいて、他の回路網に対して一つだけ用いられていないＩ／Ｏピンが、行先ロジックチップに至るパスが同様にすべて使用されているクロスバーチップとつながっている場合、部分的クロスバー相互接続は回路網を相互接続することができない。行先ロジックチップは利用可能なピンを有しているが、このような場合、Ｉ／Ｏピンはソースピンがすべて使用されている他のクロスバーにつながっており、これらのクロスバーから最初へ戻る途はない。部分的なクロスバー相互接続の容量はそのアーキテクチャーに依存している。一つの極端な例では、一つのロジックチップピンサブセットだけが存在し、一つのクロスバーがすべてのピンに作用する。このような装置は、最大の相互接続能力を有するが、非現実的な完全クロスバー接続である。他の極端な例では、サブセットサイズは、ロジックチップのピンと同数のクロスバーチップを有するものである。これは、すべての部分的クロスバーを相互接続する能力は最小であるが、依然として充分な能力を有している。極端な例の間では、各クロスバーチップが、２、３又はそれ以上の各ロジックチップのピンに作用するアーキテクチャーとなっている。クロスバーチップ数が減少し、クロスバーチップ毎のピン数が増加するにつれて、より多くの相互接続能力が利用可能となる。この変更は、以前より注目されていることではあるが、種々のクロスバーチップが作用するために、相互接続することのできない未使用ロジックチップが存在するということによるものである。クロスバーチップの数がより少なくなるとともに、

幅が広くなるにつれて、このような変更は、一般的には生じなくなる。完全クロスバーは、すべてのピンを定義されたいかなるパターンにも相互接続することができる。他の簡単な一例として、各々3個のピンを有する、参照番号1、2及び3を付した3個のロジックチップが存在し、且つ、4個の回路網A、B、C及びDが存在するものと仮定する。回路網Aはロジックチップ1及び2を接続し、回路網Bはロジックチップ1及び3を接続し、回路網Cはロジックチップ2及び3を接続し、回路網Dはロジックチップ1及び2を接続する。図8a及び8bにおいて、各ロジックチップのピンをセルの行として示しており、各クロスバーチップはクロスバーチップが作用するピン数と同数の列をカバーしている。第1のケース(図8a)では、各々1つのピンの幅を有する参照番号1、2及び3で示される3つのクロスバーチップを使用する。各クロスバーチップは、ただ1つの回路網を接続できるにすぎない。すなわち、クロスバーチップ1は、回路網Aを相互接続するようプログラムされており、クロスバーチップ2は回路網Bを接続し、クロスバーチップ3は回路網Cを接続する。未使用ロジックチップピンを利用することもできるが、回路網Dは未接続のままである。第2のケース(図8b)では、3個のピン幅を有する完全クロスバーを、クロスバーチップ1、2及び3の代わりに用いて回路網Dを接続することができる。種々の部分的クロスバー相互接続アーキテクチャによって相互接続することのできる入力設計回路網の数に基づき、アナリシス及びコンピュータモデル化を行う。結果的には、ナロウな部分的クロスバーは、ワイドなもの又は完全クロスバーと、ほぼ同じ程度に効果的である。例えば、好適実現例(14個の128ピンロジックチップ、32個の56ピンクロスバーチップ)のロジックボードに用いられている相互接続は、完全クロスバーの有する相互接続容量の98%を示している。モデリングにおいて想定されているように、実際の入力設計部が、利用可能なマルチロジックチップ回路網及びロジックチップピンの数を最大限に必要とすることは極めて稀である。実際の設計部は、ほぼ常に最大限よりも少ない回路網を有しており、上述のモデルの部分的クロスバーによって接続される回路網の平均個数よりも少ない回路網、通常かなり少ない回路網を有している。このことは、ロジック容量を保持するのに絶対的に必要であるより多くの、小さな比率のロジックチップピン及びクロスバーチップを用いることで保障され、このようにしてナロウな部分的クロスバーによって、実際の設計部がほとんど常に相互接続可能であることを保障している。ナロウなクロスバーチップは、ワイドなクロスバーチップよりかなり小さく、それ故、ピン単位では高価なものではない。

【0018】1.2.3 相互接続トライステート回路網
部分的クロスバー相互接続のようなアクティブ相互接続と実際のワイヤのようなパッシブ相互接続との重要な相違は、アクティブ相互接続が無方向性であるということである。実際、各々の相互接続は、チップ境界において金属及びトレースによって結合する一連のドライバ及びレシーバを具えている。通常の回路網は一つのドライバを有し、アクティブ相互接続で固定されたドライバ及びレシーバを用いて作成される。実際に設計する回路網の幾つかはトライステートであり、図9に示すような幾つかのトライステートドライバを有している。任意の所定時間において、最大で1個のドライバが活動状態であり、その他のドライバは回路網に対して高インピーダンスの状態にある。(伝播遅延を無視すると)すべてのレシーバは常に同一のロジックレベルにある。

【0019】1.2.3.1 トライステート回路網を積の和に置き換える
全回路網を同一のロジックチップへと区分化する場合、回路網を、積の2ステート加算、すなわち、図10にて示すような、等価なマルチプレクサで置き換えることができる。アクティブイネーブルが存在しない場合、この回路網は低ロジックレベルを出力する。時々、トライステート回路網は、受動的に高ロジックレベルにされる。必要ならば、各ANDゲートへのデータ入力を反転するとともに、最終加算ゲート出力を反転することで、イネーブルできない場合、積の和は高ロジックレベルを出力する。1以上のイネーブルがアクティブの場合、結果はすべての入力信号の加算(OR)となる。このことは容認される。その理由は、異なるデータで1以上がイネーブルされる場合、実際のトライステートドライバの動きを規定していないからである。図11a及び11bは、2種類の回路網:すなわち"フローティングハイ(floating high)"及び"フローティングロー(floating low)"を示している。リアライザシステムの設計変換システムの基本要素変換部分は、和又は積の置き換えを行う。その理由は、好適実現例のロジックチップ及びクロスバーチップとして用いられるXilinx LCAが、すべての回路網におけるトライステート駆動を一様に維持していないからである。トライステートドライバは、LCAの境界におけるすべてのI/Oピンを利用することができる。XC3000シリーズLCAの内部で利用できるトライステートドライバの数は制限されており、チップ間を結んでいる内部相互接続の数が小さいことから、各ドライバはCLBの一つの行にだけ作用する。トライステート回路網をこれらの相互接続に作成することによって、分割化に他の制約が加わり、LCAにおけるCLBの配置の自由度を制約することとなる。同時に、回路網毎に少数のドライバとトライステート接続することは、ある種のゲートアレイライブラリセルにおいては一般的なことである。結果的に、このように複雑となることを避けられる場合には、積の和の置き換えを行う。設計部を多重ロジックチップに分割化することにより、トライ

ステート回路網を２以上のロジックチップに亘って分割する場合、積の和を局所的に用いて、ロジックチップと回路網との各々の接続をロジックチップ境界における単一のドライバ及び／又はレシーバに引き下げる。図１２は、２つのドライバ及び２つのレシーバを一緒に示している。２つのドライバは局所的な積の和によって構成され、このようにして、単一のドライバ接続のみを要件として積の総和を与える。同様にして、単一のレシーバ接続を、２つのレシーバに亘って構成する。このようにして、アクティブ相互接続がなされる。トライステート回路網におけるいかなる所定の点においても、駆動“方向”はどのドライバを活動状態とするかに依存している。このことは、パッシブ相互接続と何ら差異はないが、アクティブ相互接続では、能動的に正しい方向に駆動及び受信が行われるように、アクティブ相互接続を構成しなければならない。構成によっては、このことを部分的クロスバー相互接続によって達成することができる。

【００２０】１.２.３.２　ロジック加算構成

３つの構成は、回路網を積の和に引き下げることに基づいている。ロジック加算構成は、図１３に示されているように、加算ＯＲゲートを、関連するロジックチップ中に配置する。積を発生させるＡＮＤゲートを駆動ロジックチップで構成する。この駆動チップの各々は出力チップを必要としている。各受信ロジックチップは入力ピンを必要とし、特別な場合、加算ロジックチップは各ドライバ用の入力ピンと出力ピンとを必要とする。これらの接続はすべて無方向性であり、各チップの境界に亘ってＯＢＵＦ／ＩＢＵＦ対を具えている。ドライバのピンが高価であるので、駆動ロジックチップを加算チップとして選択する必要がある。簡単のため、図中には関連するＬＣＡ基本要素をすべて示してはいない。駆動入力ピンから受信出力ピンに至る実際のパスは、ドライバのＣＬＢ及びＯＢＵＦと、クロスバーのＩＢＵＦ／ＯＢＵＦと、加算チップのＩＢＵＦ、ＣＬＢ及びＯＢＵＦと、クロスバーの他のＩＢＵＦ／ＯＢＵＦと、レシーバのＩＢＵＦとを具えている。クロスバーＩＢＵＦ遅延をＩｘとし、ロジックＣＬＢ遅延をＣ１等とした場合、全データ通路遅延は、Ｃ１＋Ｏ１＋Ｉｘ＋Ｏｘ＋Ｉ１＋Ｃ１＋Ｏ１＋Ｉｘ＋Ｏｘ＋Ｉ１である。特別な場合、すなわちロジックチップをＸＣ３０９０－７０とし、クロスバーをＸＣ２０１８－７０とした場合、遅延の総計の最大は、８２ｎｓに内部ＬＣＡ相互接続遅延を加えたものに等しい。同じ遅延がイネーブルにも当てはまる。ｎビットバスを相互接続する場合、バスの各ビットに対してすべてのイネーブルは同様のものである。この特別な構成において、駆動チップ中に積のゲートを設け、イネーブルを内部に設け、バスに必要なピンを１ビットの場合のピン数のちょうどｎ倍とする。

【００２１】１.２.３.３　クロスバー加算構成

クロスバー加算構成において、加算ＯＲゲートをクロスバーチップに配置する。この場合、図１４に示されているようなロジックを利用することのできるＬＣＡのようなＥＲＣＧＡを用いて、いくつかの例のクロスバーチップを具体化している。各ロジックチップは、ドライバとしての１ピン及び／又はレシーバとして１ピンを必要としている。クロスバーチップは、加算ゲートのための１以上のロジック素子を有している必要がある。クロスバー加算とは、ロジックチップ中のロジックをすべて用い、クロスバーチップ中のロジックを全く用いず実行するということではない。重要な相違点は、クロスバーチップに配置されたロジックが、実現される設計ロジックの一部ではないということである。ロジックは、単に、トライステート回路網の相互接続機能を達成する役割を果たすに過ぎない。この構成では、２以上の駆動ロジックチップを設けた場合、従来よりも使用するピンの数が少ない。ｎビットバスはピンのｎ倍も作用する。全遅延は、Ｃ１＋Ｏ１＋Ｉｘ＋Ｃｘ＋Ｏｘ＋Ｉ１、すなわち、最大５１ｎｓにまで引き下げられる。イネーブルも同じ遅延を有する。

【００２２】１.２.３.４　双方向性クロスバー加算構成

図１５にて示されているように、クロスバーチップの加算ゲートを、双方向性クロスバー構成における双方向性接続を介して連絡している。ＯＲゲートへのパスをイネーブルできるＡＮＤゲートを、クロスバーチップ中に設け、フィードバックラッチアップパスをブロック化する。ロジックチップは、レシーバのみの場合には１つのピンを必要とし、ドライバ又は、レシーバ及びドライバ双方の場合には２つのピンを必要とする。この２つのピンの一方は信号自体のためのものであり、もう一方はイネーブル出力のためのものであり、これはクロスバーチップに用いられる。１ビット以上の信号イネーブルを用い、マルチビットバスによって、相互接続を減少させることができる。同一のクロスバーチップを介して、１ビット以上のバスを相互接続する場合、１セットのイネーブル信号をチップに供給する必要がある。好適なＬＣＡ例においては、全データ通路遅延を、Ｏ１＋Ｉｘ＋Ｃｘ＋Ｏｘ＋Ｉ１、すなわち、４２ｎｓとしている。積の和が、２以上のＣＬＢを用いる場合、付加的なＣｘ（１０ｎｓ）を加えることができる。イネーブル遅延は出力遅延Ｏ１ではなくて、ＯＢＵＦＺのイネーブル遅延Ｅ１に依存している。

【００２３】１.２.３.５　双方向性クロスバートライステート構成

これまで説明したすべての構成を同一のハードウエアで使用することができることに注意しなければならない。基本要素の配置及び相互接続のみが変化する。最終的に、クロスバーチップが内部トライステートを維持する場合、図１６にて示すように、双方向性クロスバートラ

イステート構成によって、クロスバーチップ内部の実際のトライステート回路網が二重になる。各ロジックチップの実際のトライステートドライバは、クロスバーチップのバスにそのまま伝えられる。このことは、イネーブル信号の相互接続によって達成されるはずである。ドライバがイネーブルされていない場合、クロスバーチップのバスを駆動させる。ＬＣＡをクロスバーチップとして用いる場合、上記内部トライステート相互接続を用いる。特に、ロジックチップの境界にＩＢＵＦ／ＯＢＵＦＺ対、クロスバーチップ境界に各ロジックチップの他のＩＢＵＦ／ＯＢＵＦＺ対、各ロジックチップにＴＢＵＦを設け、内部トライステートラインを駆動する。各イネーブルはＯＢＵＦ及びＩＢＵＦを通過する。イネーブルされたデータパス遅延の総計は、Ｏ１＋Ｉｘ＋Ｔｘ＋Ｏｘ＋Ｉ１、すなわち、３９ｎｓ（ＸＣ３０３０－７０ＬＣＡクロスバー）であり、イネーブル遅延の総計は、Ｏ１＋Ｉｘ＋ＴＥｘ＋Ｏｘ＋Ｉ１、すなわち４５ｎｓである。以前のように、２ビット以上のバスを、同一のクロスバーチップを介して相互接続する場合、１セットのイネーブル信号のみを、チップに供給する必要がある。この構成では、クロスバーチップをＬＣＡ又は内部トライステート機能を有するＥＲＣＧＡとする必要があり、これら内部相互接続の利用を条件としている。特に、XC2000シリーズは、内部トライステートを有していないが、XC3000パーツは有している。XC3030は、８０個のＩ／Ｏピン、１００個のＣＬＢ及び２０個のトライステート駆動可能内部‘ロングライン’を有している。このようにして、最大で２０個のこのようなトライステート回路網を、この構成中の一つのクロスバーチップによって相互接続し得る。これは、相互接続の限界となり得るが、いろいろな場合のほんの一部にすぎず、Ｉ／Ｏピンの限界を与えるものである。現在のところ、XC3030はXC2018の２倍高価である。ハードウエアにトライステート構成を用いる場合、他の構成は妨げとはならず同様に使用することができる。

【００２４】１.２.３.６　すべての構成の概要

このチャートは、構成を要約したものである。

	ロジック加算	クロスバー加算	双方向性クロスバー加算	双方向性クロスバートライステート
ピン／ロジックチップ双方向性	＝駆動＋受信	2	1データバス 1共有可能イネーブル	1データバス 1共有可能イネーブル
駆動のみ	第1チップ：0 その他：2	1	1データバス	1データバス
受信のみ	第1非加算：2 その他：1	1	1共有可能イネーブル	1共有イネーブル

遅延

（LCAクロスバーチップ：＋LCA相互接続、70MHz　LCAチップスピードであると仮定する）

データバス	82ns	51	42	39
イネーブル	82	51	46	45

チップ毎のリソース

（dはドライバの数）

駆動のみ	AND中に2個	AND中に2個	0	0
受信のみ	0	0	0	0
双方向性	AND中に2個	AND中に2個	0	0
クロスバー	0	OR中にd個	OR中にd個	d個のTBUF

明らかに、ロジック加算構成は有効ではない。クロスバー加算は、かなり高速で、少数のピンを使用し、多くの場合シンプルである。双方向性クロスバー加算は、依然わずかに高速であり、双方向性バスのピン数を減少させる可能性を有しているが、かなり複雑であり、クロスバーチップに限られたロジックリソースをより必要とする。トライステート構成によって、同様のピンを必要とし、及び遅延が生じるが、より高価なクロスバーチップを必要とする。

【0025】1.2.3.7　普通のクロスバー加算構成と双方向性クロスバー加算構成との比較

最も効果的な構成の特性をテストすることは有益である。以下の表は、普通及び双方向性のクロスバー加算構成を用い、多数の双方向性回路網を相互接続し、且つ、LCAをクロスバーチップとして用いる場合に生じるクロスバーCLBの数とクロスバーCLB遅延とを示している。72個のI／Oピンを有し、100個のCLBを用いることのできる、XC2018-70クロスバーチップを用いるものと仮定する。各々のCLBは4個までの入力端子及び2個までの出力端子をサポートする。各ロジックチップがイネーブルを共有せず、回路網と双方向性の接続を有し、各テストにおいて、クロスバーチップの72個のI／Oピンすべてを用いるものと仮定する。

	クロスバー	双方向性クロスバー
	加算	加算
１８個の双方向性回路網サービング	９ ＣＬＢｓ	１８ ＣＬＢｓ
２個のロジックチップの各々	１ Ｃｘ	１ Ｃｘ
１２個の双方向性回路網サービング	１２ ＣＬＢｓ	２４ ＣＬＢｓ
３個のロジックチップの各々	１ Ｃｘ	２ Ｃｘ
９個の双方向性回路網サービング	９ ＣＬＢｓ	２７ ＣＬＢｓ
４個のロジックチップの各々	１ Ｃｘ	２ Ｃｘ
６個の双方向性回路網サービング	１２ ＣＬＢｓ	２４ ＣＬＢｓ
６個のロジックチップの各々	２ Ｃｘ	２ Ｃｘ
３個の双方向性回路網サービング	１２ ＣＬＢｓ	３０ ＣＬＢｓ
１２個のロジックチップの各々	２ Ｃｘ	３ Ｃｘ

双方向クロスバー加算構成はＣＬＢを２．５倍まで使用し、クロスバーチップがルートしない可能性、すなわち内部相互接続遅延が大きくなる可能性が増大する。しかし、依然として１００個のＣＬＢが使用可能となるまでには程遠い。代わりに、無方向性構成では、ロジックチップを特別なゲートを操作するのに好適な位置に設けるが、ロジックチップにかなり多くのゲートを設けている。双方向性構成では、特別なＣｘ遅延がしばしば生じそのスピードの利点を相殺してしまう。リアライザシステムの好適例では、トライステート回路網のためのクロスバー加算構成を用いている。

【００２６】１.２.４　システムレベル相互接続

クロスバーチップによって相互接続された１セットのロジックチップのパッケージ化は、単一の回路ボードにおいて行うのが一般的である。システムが大規模すぎて単一のボードに適合しない場合には、システムレベル相互接続を用いて、ボードを同じように相互接続しなければならない。極めて広域のパス配線のために、２以上の回路ボードに亘る単一の部分的クロスバー相互接続及びロジックチップを拡張することは非実用的である。例えば、３２個の１２８ピンロジックチップと６４ピンのクロスバーチップとの複合体を、２つのボードに、それぞれ１６個のロジックチップと３２個のクロスバーとに分割するものと仮定する。複合体を、ロジックチップとクロスバーチップとの間で切断する場合、背面接続の対を介して、ロジックチップとクロスバーチップとの間に総計で４０９６個の相互接続を行う必要がある。これとは別の方法で、‘中間’で、すなわち、１６個のロジックチップ及び３２個のクロスバーチップで各ボード毎に切断する場合、ボード１のロジックチップをボード２のクロスバーに接続するパス（１６個のロジック×６４個のピン＝１０２４）及びその逆のパス（もう一つの１０２４、合計で２０４８）の全てをクロスさせなければならない。このような単一の相互接続では発展の可能性がないといった他の制約もある。定義によれば、各クロスバーチップは全てのロジックチップと接続している。特定数のロジックチップで構成する場合、それ以上を加えることができない。その代わりに、回路基板上に一緒にパッケージ化することのできるロジックチップとクロスバーチップとの最大規模の複合体をロジックボードと称し、モジュールとして用い、これらの多数をシステムレベル相互接続によって接続する。２以上のボードに及ぶ相互接続回路網を提供するために、各ロジックボードのクロスバーチップの各々の付加的なＩ／Ｏピンに対して、ボードから離れた付加的な接続を行い、ロジックボードＩ／Ｏピンを確立する（図１７）。ロジックボードＩ／Ｏピンに接続するのに用いられるクロスバーチップＩ／Ｏピンは、ボードのロジックチップＩ／Ｏピンと接続しているものとは別のものである。

【００２７】１.２.４.１　部分的クロスバーシステムレベル相互接続

ロジックボードを相互接続するための一つの手段では、部分的なクロスバー相互接続を再び適用し、各ボードをロジックチップの如く取り扱うとともに、付加的なクロスバーチップのセットを用いてボードのＩ／Ｏピンを相互接続する。この部分的なクロスバーは、ボックス中のすべてのボードを相互接続する。第３番目の相互接続を、ラック中のすべてのボックス等を相互接続することに再び適用する。終始同一の相互接続方法を適用することによって、概念の簡易化及びボードレベル相互接続との一体化といった利点が得られる。リアライザシステム中のクロスバーチップを区別するために、ロジックチップを相互接続している部分的クロスバー相互接続をＸレベル相互接続と称し、そのクロスバーチップをＸチップと称する。ロジックボードを相互接続している相互接続をＹレベル相互接続と称し、そのクロスバーチップをＹチップと称する。Ｘレベル相互接続では、各ロジックボードの分割と同様の分割を用いて、各ロジックボードのＩ／Ｏピンを適切なサブセットに分割する。各Ｙチップのピンを、すべてのロジックボードの各々からのピンの同一のサブセットに接続する。サブセットと同数のＹチップを使用する。各Ｙチップは、サブセットのピン数とロジックボードのピン数とをかけ算した結果と同数のピンを有している。同様にして、各Ｙチップの付加的なＩ／Ｏピンにボックスから離れた付加的な接続を行い、ボックスＩ／Ｏピンを構成する。この各々を、各ボックス

における分割と同様の分割方法を用いて適切なサブセットに分割する（図１８）。各Ｚチップのピンを、各ボックスからのピンの同一のサブセットに接続する。サブセットと同数のＺチップを用いる。各Ｚチップは、サブセットのピン数とボックスの数とのかけ算した結果と同数のピンを有している。部分的なクロスバー相互接続の付加的なレベルを構成する方法を、必要である限り継続する。入力設計部を区分化する場合、ロジックチップのＩ／Ｏピンの数が限定されているように、ボード上及びボードから離れて配線されている回路網がボードのＩ／Ｏピンを介しており、ボードのＩ／Ｏピンの数の限定には一定の制約があることがわかる。多重ボックスリアライザシステムにおいて、ボックスＩ／Ｏピンの数が限定されていること等がわかる。設計メモリのような特別の機能が付随する場合を除き、チップ、ボード又はカートリッヂに関する配置を最適にするための相互接続シンメトリー手段は必ずしも必要ではない。双方向性回路網及びバスを、クロスバー加算方法のようなトライステートセクションにおいて説明した方法のうちの一つを用いて具体化する。この方法は、回路網がつながっている相互接続階層の各レベルに亘って適用される。好適な具体例は次のとおりである。

・全ハードウエアシステムに亘る三つのレベルにおいて、部分的クロスバー相互接続を階層的に用いる。

・ロジックボードが、各々１２８個の相互接続されたＩ／Ｏピンを有する、最大１４個のロジックチップと３２個のＸチップから成るＸレベル部分的クロスバーとを具えている。各Ｘチップは、各々１４個（全５６個）のＬチップへつながっている４個のパスと２個のＹチップの各々へつながっている８個のパスとを具え、全体として、ボード毎に５１２個のロジックボードＩ／Ｏピンを具えている。

・１個のボックスが、各々５１２個の相互接続されたＩ／Ｏピンを有する１～８個のボードと６４個のＹチップから成るＹレベル部分的クロスバーとを具えている。各Ｙチップは、ロジックボードＩ／Ｏピンを介して各ボードのＸチップにつながっている８個のパスと１個のＺチップにつながっている８個のパスとを具えており、合計でボックス当り５１２個のボックスＩ／Ｏピンを具えている。

・ラッチは、各々５１２個の相互接続Ｉ／Ｏピンを有する。１～８個のボックスと６４個のＺチップから成るＺレベルクロスバーとを具えている。各Ｚチップは、ボックスＩ／Ｏピンを介して、各ボックス中のＹチップにつながっている８個のパスを具えている。

【００２８】１.２.４.２　双方向性バスシステムレベル相互接続

コンピュータハードウエアの実行には、双方向性バスの背面を用いての、ロジックボードのシステムレベル相互接続に関する他の方法が必要となる。以前と同様、各ロジックボードにＩ／Ｏピンを設け、各ボードのＩ／Ｏピンを、バスワイヤによって、他のすべてのボードの同じＩ／Ｏピンに接続する（図１９）。いくつかのロジックボードＩ／Ｏピンは無駄である。すなわち、設計回路網に対して相互接続不能である。その理由は、一つの設計回路網を相互接続するためのバスワイヤを使用することによって、バスを共有している他のすべてのボードのバスワイヤに接続されたピンを使用できなくしてしまうからである。相互接続することのできる設計回路網の最大数は、バスワイヤの数、すなわち、ボード毎のＩ／Ｏピンの数と等しい。特別の場合として、８個のボードにおいて、一つの共通相互接続バスを、各ボードの５１２個のＩ／Ｏピンを接続している５１２個のバスワイヤが共有している（図２０）。２番目、３番目、４番目、５番目、６番目、７番目及び８番目ボードの回路網が異なる配線であると仮定すると、解析により、各ボードと接続している回路網の平均数は各々の場合５１２であり、すべての回路網において、ボード及びバスは１１６６個のピン幅まで許容されるはずであることがわかる。このことは、単一の背面のボード数を小さくし続けることによって部分的に軽減される。しかし一組の双方向性バスと相互接続されたボードの最大数は制限されている。大規模システムをより効果的に構成するためには、バスのグループを階層的に相互接続する。図２１に示されている第１の例では、各々４個のボードを接続している２組のバスＸ０及びＸ１を有している。Ｘレベルのバスを、他のバスＹで相互接続する。Ｘバス中の各々のワイヤを、再構成可能な双方向性トランシーバによってＹの片方に接続する。双方向性トランシーバの構成によって、Ｘ及びＹのワイヤが絶縁されているかどうか、ＸがＹを駆動又はＹがＸを駆動するかどうかが決定される。回路網が、左側のボードの組又は右側のボードの組のみを接続する場合、Ｘレベルバスの一方又は他方のみを用いる。両側にボードを具えている場合、Ｘ０及びＸ１のワイヤを各々使用し、これらのワイヤをトランシーバを介してＹのワイヤで相互接続する。各ボードは、Ｘレベルバスの一方の幅と同数のＩ／Ｏピンを有している必要がある。Ｙを介しての相互接続が双方向性、すなわち、Ｘ０又はＸ１のいずれか一方によって駆動される場合、追加的な信号がＸ０及びＸ１から流れ、トランシーバの方向性を動的に制御する。この相互接続を分析しボード間の回路網を相互接続する機能を示す。この際、上記と同じ回路網ピン数及びＩ／Ｏピン数であると仮定する。シングルレベル方法では、全回路網の総計と同じ幅を必要とするが、これを２つに分割し、必要とされる最大幅を１０％～１５％に減少させている。階層は、最大でも、バス当りただ２つのボード又は２つのグループのボードを有するのみである（図２２）。双方向性バス相互接続は簡素であり、組立てが容易であるが高価である。その理由は、かなり多くのロジックボードＩ／Ｏピンを他のボ

ードの回路網に接続することによって無駄にしているからである。このことを避けるために、階層化及び短絡背面を導入しても、効果が極めて小さいことが証明されている。更に、双方向性トランシーバを導入することによって、シングルレベル背面バス相互接続が部分的クロスバーよりも優位にあるスピード及びコストの面での利点を除去してしまう。結果的に、好適例のシステムレベル相互接続に部分的クロスバーを用いる。

【００２９】1.3　特定目的の構成素子

特定目的の構成素子とは、入力設計を実現し、好適例のロジックボードのＬチップの位置に取り付けるハードウエア構成素子であるが、ロジックチップを構成する組合せロジックゲート又はフリップフロップではない。

【００３０】1.3.1　設計部メモリ

多くの入力設計部はメモリを具えている。ロジックチップがメモリを具えているならば理想的である。電流ロジックチップデバイスはメモリを具えていない。メモリを具えるとするとメガバイト規模のメインメモリを依然として必要とし、これは、ロジックチップには決して望めないことである。従って、設計メモリデバイスを、リアライザシステム中に設けることとする。

【００３１】1.3.1.1　設計部のメモリアーキテクチャ

設計部メモリモジュールのアーキテクチャを以下の要件に基づき構成する。：

ａ）設計部のメモリモジュールは、設計部の一部であるため、他の構成要素と自由に相互接続できるようにする必要がある。

ｂ）ロジックチップと同様に効果的な相互接続を行うことができるように、データ、アドレス及び制御入出力の割り当てに自由度を設け、バスの相互接続を行う必要がある。

ｃ）種々の容量及びビット幅を有する１以上の設計部メモリを実現できる構成の変更を可能にする必要がある。

ｄ）ホストインタフェースが、設計部とデバッガタイプの対話ができるように、アクセス可能である必要がある。

ｅ）メモリモジュールはダイナミックではなくスタティックである必要がある。これによって、設計部を意のままに、ストップ、スタート又は任意のクロックスピードでランさせることができる。

これらの要件を満足するメモリモジュールの一般的アーキテクチャーを図２３に示す。設計部との相互接続可能性及びリアライザシステムの物理的構成に関する柔軟性を維持するために、置き換えたロジックチップと同一の相互接続及び他のピンに接続されたＬチップソケットにプラグで接続するように、メモリモジュールを設計する。必要なだけのモジュールを取り付ける。ＲＡＭチップを相互接続に直接接続しない。その理由は、主にチップのデータ、アドレス及び制御機能を特定のピンに定めているからである。部分的クロスバー相互接続の成功が、自由にＩ／Ｏピンとの内部相互接続を割り当てることのできるロジックチップの機能に依存しているために、ロジックチップの場所に配置されたノンロジックチップデバイスは、同様の機能を有している必要がある。このことを達成するとともに、メモリモジュールに他のロジック機能を提供するために、ロジックチップをメモリモジュール中に取り付け、ＲＡＭチップをクロスバーのＸチップと相互接続する。特定のＲＡＭピンを、任意に選択されたＸチップピンと相互接続し、メモリモジュールを構成する。この際、メモリモジュールを使用する場所に、ロジックチップが使用するのと同一のＬ－Ｘパスを使用する。１個よりも多くのロジックチップをモジュール毎に使用する。その理由は、接続すべきＲＡＭピン及びＬ－Ｘパスの数が多いからである。メモリモジュールのロジックチップによって、メモリモジュールに構成可能性及びホストアクセス可能性を提供する。ロジックチップを介して、種々の容量、ビット幅及び入力／出力構造を有するＲＡＭチップを接続するように、アドレス、データ及び制御パスを構成する。メモリモジュールを、１個の大規模メモリ又は幾つかの小規模メモリで構成することができる。これらのロジックチップの各々をホストインタフェースバスに接続するとともに、バスインタフェースロジックをロジックチップ中に構成することによって、ホストプロセッサが、ＲＡＭをランダムにアクセスすることができる機能を実現する。これによって、デバッガのようなコンピュータプログラムを用いて、メモリ内容を検査及び修正する。これらのロジック構造の具体例を、以下に示す。実現する設計部のタイミングに関する要件を満足する、入手可能で高密度且つ安価なスタティックメモリを、設計部メモリとして選択する。好適例では、このようなデバイスを富士通MB84256のような８ビット３２ＫのCMOSSRAMとしている。これによれば、スピードを５０ｎｓに落とすことができる。かなり高速のデバイスを用いればリターンを減少させることができる。その理由は、リアライザシステムのクロスバーチップ相互接続遅延が主な原因となり始めるからである。ダイナミックメモリデバイスを用いてはいない。その理由は、ダイナミックメモリデバイスではこれらを規則的にリフレッシュしなければならず、リアライザシステムに種々の問題が生じる。入力設計部にダイナミックメモリが必要な場合は、入力設計部はおそらくリフレッシュロジックを具えている。しかしながら、実現された設計部が１００％の設計スピードで動作できないので、設計部をリフレッシュさせることは成功しない。実際、デバッギングの際に、設計部の実行を停止させることが望ましい。すなわち、設計部はシステムの一部分であり、リフレッシュをするためには入力設計部に具わっていない他のいくつかの構成要素に依存しなければならない。つまり、設計部にスタティックメモリを必要とす

る場合、ダイナミック設計メモリのリフレッシュを行うことは非現実的である。スタティックメモリによればリフレッシュサイクルを無視できるため、ダイナミックメモリを設計部内に実現することができる。このようにして、スタティックデバイスを用いて設計部メモリを具体化する。

【００３２】１.３.１.２　ロジックチップをＲＡＭとクロスバーとの相互接続に使用する

理想的には、単一のロジックチップを用いてＲＡＭとＸレベルクロスバーとを相互接続する。この際、すべてのＬ－Ｘ相互接続パスと同様に、すべてのＲＡＭ信号ピンを接続するのに十分なピンを用いる。実用的なリアライザシステムメモリモジュールでは、単一のロジックチップが実行困難な程多くのピンを必要としている。例えば８個の３２Ｋ、８ビットＲＡＭから成る２つのバンクを、１２８個のＬ－Ｘパスを有するモジュール中に用いるものと仮定する。各々のＲＡＭバンクは、１５個のアドレスピンと、８個の書込みイネーブルピンと、６４個のデータピンとを具えている。２個のバンク及びＬ－Ｘパスは、３０２個のピンと、ホストインタフェースバスのためのピンとを必要としている。これは、使用可能なロジックチップのピン数の２倍である。１より多くのロジックチップを用いなければならない。ここで述べたアーキテクチャでは、多くの小型ロジックチップを用いており、これらのチップにはアドレス、コントロール及びデータパスに関する特別の機能が与えられる。

【００３３】１.３.１.２.１　メモリアドレスロジックチップ

図２３において、アドレス及びコントロールロジックチップを、“ＭＡ０”及び“ＭＡ１”で示している。ＲＡＭをバンクに分割する。このバンクを各々のＭＡチップで制御する。モジュールによって実現すべき分離設計部メモリの最大数と同数のＭＡチップを設ける。その各々に、クロスバーとつながっているＬ－Ｘパスのセット、すなわち、バンクのアドレス及びコントロールラインにとって必要なだけのパスを設ける。ＭＡ０及びＭＡ１は別のパスの組を使用する。例えば、各々ＲＡＭの半分に接続されている２個のＭＡチップによって、２個の独立メモリを実現することができる。１個の大型メモリを実現する場合、両方のＬ－Ｘパスの組を用いて、アドレス及びコントロール回路網を両ＭＡチップに接続する。各ＭＡチップはバンク内の全ＲＡＭのアドレス入力を制御する。アドレス入力を、単一のバスで結びつける。各々のＭＡチップは、個々で、ＲＡＭへの制御入力を制御し、データをアドレス指定したＲＡＭにのみ書き込むことができるようにしている。つまり、各ＭＡチップをアクセス可能とするために、ホストインタフェースバスに接続するとともに、このメモリモジュールのすべてのロジックチップと共通なコントロールバスに接続している。図２４は、いかにしてＭＡチップをＸレベルクロスバー及びＲＡＭチップに接続するかを、さらに詳細に示している。図に示すように、ロジック及びデータパスに従ってＭＡチップを構成する。すべてのアドレスがクロスバーからＭＡチップに入る。通常、（バスインタフェースを非活動状態とした場合）、ＲＡＭアドレスビットの数に相当するアドレスビットの部分をパスし、ＭＡチップによって制御されるバンク中のＲＡＭをアドレス指定する。その他のアドレスビット及び設計部の書き込みイネーブルによって、各々のＲＡＭの書き込みイネーブル信号を制御するデコーダロジックを駆動する。この設計部メモリに必要とされる構成に応じて、ロジックを構成する。例えば、設計部メモリが１個のＲＡＭと同じビット幅を有し、設計部が書き込みイネーブルを主張する場合、アドレスビットに従い、ただ一つのＲＡＭ書込みイネーブルが主張される。設計部メモリが１個のチップの２倍の幅を有する場合、一対のＲＡＭ書込みイネーブルが主張される。メモリのデータパス幅のサブセットを各々制御している１より多くの書込みイネーブルを有する設計部メモリを望む場合、幾つかの設計書込みイネーブル回路網を用いることができる。各回路網は、ＭＡ及びＭＤチップ中のデコードロジックの構成を適切なものとし、上述のラインに沿って作動する。このことは、ＭＡチップへつながっているＬ－Ｘパスと、ＭＤチップへつながっているコントロールバスパスとの使用可能度に依存している。バスインタフェースロジックによって、ホストインタフェースバスを介し、ホストはこのＲＡＭをアクセスさせることができる。この組となっているＲＡＭをバスを用いてアドレス指定する場合、バスインタフェースは、アドレスマルチプレクサ（‘ｍｕｘ’）を切り換え、ＲＡＭをそのアドレスにアドレス指定する。ホストが１個のＲＡＭに書込みを行う場合、バスインタフェースロジックは、信号をデコーダロジックに送信する。デコーダロジックは、アドレスビットを使用し、ＲＡＭを駆動せずに、適切なＲＡＭ書込みイネーブルを主張する。最終的には、ＭＤチップ中のデータパスを制御するのに、幾つかの信号を必要とする。ＭＤチップのすべてを、ＭＤチップと同一のＬ－Ｘパスに接続してはいないので、ＭＤチップは、設計部からのアドレス及びコントロール信号をアクセスする必要はない。コントロールバスをすべてのＭＡ及びＭＤチップに接続し、これらの信号及びバスインタフェースコントロール信号をＭＤチップへ送信できるようにしている。

【００３４】１.３.１.２.２　メモリデータパスロジックチップ

ＭＤチップは、ビットスライス構成に従ってデータパスを操作する。クロスバーと交差してビットスライスを行うことによって、リアライザシステム中のマルチビットバスデータパスを相互接続する。チップ毎に１又は２ビットを用いて、バスはＸチップと交差して広がっている。ＭＤチップをビットスライスし、これらのバスとの

接続を容易にしている。各ＭＤチップをすべてのバンク中の各ＲＡＭ中の同一のビットに接続するとともに、Ｘチップのサブセットに接続する。同一のＲＡＭビットのすべてをＭＤチップ中で結びつけ、種々のビット幅及びサイズ設計部メモリの構成に柔軟性を持たせることができる。ＭＤチップ中にロジック及びデータパスを適切に構成することによって、ＲＡＭ幅の種々の倍数で、設計部メモリを構成する。‘ｎ’個のＭＤチップ及び‘Ｍ’個のＸチップを設ける場合、各ＭＤチップをＭ／ｎの種々のＸチップを用いて接続する。各データビットは、２個のＬ－Ｘパス、すなわち、クロスバー加算相互接続構成のために、分離Ｉ／Ｏ構成のためのＤＩ及びＤＯパス又は共通Ｉ／Ｏ双方向性構成のための加算入力及び加算結果のいずれか一方である。このようにして、各ＭＤチップは少なくとも２＊Ｍ／ｎ個のＬ－Ｘパスを有している。これらに加えて、付加的なパスを設けることができる。これら付加的なパスを、ＭＡのＬ－Ｘパスに重ねることができる。ＭＤチップ、ＲＡＭ及びＲＡＭビット幅の数を選択し、これらの制約及び容量制約を適合させ、ＭＤチップに用いられるロジックチップ中のピンの数を有効に用い、これを偶数となるようにしている。工業規格スタティックＲＡＭチップは、双方向性データピン（ＤＱと称する）を有する共通Ｉ／Ｏ構造を有しており、データイン及びデータアウトに用いられる。これは、アドレス入力ピン（ＡＤＤＲ）及び書込みイネーブルピン（ＷＥ）を有している。この実現例において、出力イネーブルピン及びチップ選択ピンは永続的にイネーブルされており、出力ピンは書込みイネーブルで制御する。必要な場合にはＲＡＭの読出しを行い、アドレスデータをＤＱピンで駆動する。書込みイネーブルを主張する場合、データインをＤＱピンで受信する。この主張の終了時に、データをアドレスロケーションに書込む。規格デバイスは、書込みイネーブルの終了時にセットアップ中のデータのみを必要とし、また、ゼロ保持時間を必要とし、これによってデータパスの書込みイネーブル制御を可能としている。設計部メモリが共通Ｉ／Ｏを必要とする場合、設計部はトライステート回路網となる。これは、クロスバー加算構成を用いて実現される。すなわち、駆動ピンはそのイネーブルによって、別々にゲートされ、受信ピンを駆動する加算ＯＲゲートに集められる。ＲＡＭＤＱデータピンを、図２５に示されているようなＭＤチップ中に構成されるロジック及びデータパスによってインタフェースさせる。（一つのビットすなわちビット‘ｎ’を図示する。他も同様である。）Ｌチップがトライステートドライバを有している場合に、Ｘチップ中の加算ゲートを駆動するイネーブルゲートを有しているように、Ｘチップ中の加算ゲートを駆動するイネーブルゲートを用いて各ＭＤチップを構成する（ＭＤ‘ｎ’を図示する）。設計部メモリ入力回路網によって、出力端子をイネーブルするとともに書込みをディゼーブルする場合、ロジックは、ＲＡＭの加算ゲートへの出力をゲートするとともに、受信ドライバをディゼーブルする。もし、そうでなければ、回路の値は加算ゲートからＲＡＭへと伝達され、書込みイネーブルが主張されると書込みが可能となる。上述したように、設計部書込みイネーブル及び出力イネーブル信号がＭＡチップから（コントロールバスを介して）生じることに注意しなければならない。バスインタフェースロジックは図示していない。設計部メモリが分離Ｉ／Ｏを必要とする場合、これは、図２６に示されているように、ＳＲＡＭの共通Ｉ／Ｏから抽出される。出力イネーブルが主張される場合、データアウトは、常にＳＲＡＭのデータピンステートを反映している。書込みイネーブルが主張される場合、データインはＳＲＡＭのＤＱピンに伝達される。上記の図面では、設計部データビットに接続された１個のＲＡＭのみを図示している。時には、数個のＲＡＭを設けており、この場合、設計部メモリ中のロケーションの数は、単一のＲＡＭチップの大きさの倍数となっている。このような場合、図２７に示しているように、ＭＤチップを構成する。幾つかのＲＡＭ各々のＤＱピンをこのＭＤチップに接続する。ローアドレスビットと設計部及びバスインタフェース制御信号とが、コントロールバスを介して、ＭＡチップからＭＤチップへと伝達される。読出しの場合、アドレスのロービットは、マルチプレクサを介してＲＡＭＤＱ出力のいずれか一つを選択する。選択された出力は設計部出力イネーブルによってゲートされ、前述の例と同様に設計部メモリデータアウトを構成する。設計部がその出力イネーブルを主張する場合、ドライバをイネーブルすることによって、データインはＲＡＭＤＱ入力の内のいずれか一つに伝達される。ローアドレスビット及び設計部書込みイネーブル信号によって駆動されるデコードロジックによって、駆動すべき適切なドライバを選択する。ＲＡＭチップの書込みイネーブルをＭＡチップによって駆動することを中止する。図２７は、分離Ｉ／Ｏ構成を示している。共通Ｉ／Ｏ構成は、クロスバー加算ゲートによって駆動されるデータインと、設計部出力イネーブル及び書込みイネーブルによってゲートされるデータアウトとに類似しており、図２５に示すように加算ゲート入力を駆動する。ホストインタフェースが、ホストインタフェースバスを介してこのメモリをアクセスする場合、ＭＡチップによって構成されるロジックは、バスをアクセスするための制御信号を出力する。この信号は、コントロールバスを介してＭＡから伝達される。バスが読出しを行う場合、バス読出しイネーブルは、マルチプレクサがアドレス指定したＲＡＭより選択したデータを、このＭＤチップに対応するホストインタフェースバスデータビットに伝達する。バスが書込みを行う場合、バスデータビットからのデータを他のマルチプレクサを用いてドライバにスイッチする。このデータは、通常の書込みと同じプロセスに

よって選択されたＲＡＭのＤＱピンへ伝達される。この説明は、単一の設計部メモリのデータパス幅から単一のデータビットを用いて構成した、ＭＤチップ構成を示していることに注意しなければならない。設計部メモリ構成によるものであり、且つモジュール中のＭＤ及びＲＡＭチップの数を必要とする場合、単にデータパスを適切に曲げることによって、１より多くのデータビットが各ＭＤチップ中に現れる。さらに、前記データパス及びコントロールラインを曲げることによって、一組の共通ＭＤチップを用い、１より多くの設計メモリを実現し、幾つかのメモリを具体化する。メモリモジュールにつながっているあるＬ－ＸパスをＭＡチップにのみ接続し、且つあるＬ－ＸパスをＭＤチップにのみ接続しているので、適切なＬ－Ｘパスを用いて設計部メモリに接続された回路網を相互接続するためだけに設計部変換相互接続プロセスを組立てる。

【００３５】１.３.１.３　設計部メモリのための設計変換

オリジナル設計ファイル中で利用可能な構成のいずれか一つに対応する設計部メモリＲＡＭ基本要素を用い、入力設計部中の設計部メモリを特定する。設計部変換方法は一組の予め定義した部分的ネットリストファイルに基づくものである。この内の一つはメモリモジュールのロジックチップの各々のためのものであり、上で示したように、特別のメモリ構成をするために構成すべきすべてのロジック及びデータパスに関するステートメントを用いている。予め定義されたファイルは、相互接続を用いて、設計部メモリアドレス、データ及びコントロール接続を行うのに用いられるモジュールＩ／ＯピンのＩ／Ｏピン数仕様を除いて、完全なものである。この方法は以下のとおりである。：設計変換のセクションにて述べるように、以下に示すような設計部メモリに対して特別な例外があるものの、一般的な方法を設計変換に用いる：

・設計部リーダは、特定のベクトルメモリに対するメモリ基本要素を、設計データ構造に読出す。どの構成を用いるかを特定するためのデータを、メモリのデータ構造レコード中に記録する。

・変換ステージが、構成を利用可能であり、且つ、ピンが構成と正しく対応していることをチェックする。

・ユーザは、どのボード上のどのＬチップ位置にメモリモジュールが搭載されているかをパーティショナ（partitioner）に告げる。このデータに基づき、パーティショナは、一般的分割化アルゴリズムに従って、記憶のためのメモリモジュールを選択する。択一的に、ユーザは、このデータをオリジナル設計ファイル中の基本要素と関連づけることによって、メモリを特定のモジュールに割り当てることができる。設計部リーダは、メモリの基本要素レコード中に、オリジナル設計ファイルを具えている。

・次に、インタコネクタは、メモリに接続された回路網及びピンを、特定のＬ－Ｘ相互接続パスに割当てる。アドレス及びコントロール回路網をＭＡチップに接続している特定のパスにのみ割当て、且つ、データ回路網をＭＤチップと接続しているパスにのみ割当てることができるという制約を条件として、インタコネクタはパスの割当てを行う。各クロスバーチップセットの回路網相互接続能力を決定する場合、これらのセットを拒否する場合、及び必要とされるＭＡ又はＭＤチップを接続していないパスを得られない、又は使用することができない場合、相互接続を行う際に、これらの制約を適用する。

・リアライザシステム中の各ロジックチップに関するネットリストファイルに書込みを行う場合、各々の設計部メモリ回路網接続は：

１）ＭＡ又はＭＤのいずれかを相互接続手続によって、基本要素が選択するパスに接続するかを決定すること

２）通常のロジックチップＩ／Ｏピン数を得る場合に説明したのと同様の手続を用い、パス数とＭＡ／ＭＤチップ数とからロジックチップＩ／Ｏピン数を得ること

３）これまで他の回路網に割当てられていないこのＭＡ／ＭＤチップの回路網にからの、予め定義されたアドレス、データ又は制御接続を選択すること

４）ステートメントを、このロジックチップのネットリストファイルに加え、このロジックチップＩ／Ｏピン数を、予め定義した設計部メモリ接続に接続するのに用いることを明示することによって、ネットリストされる。

・ネットリストファイルを、ネットリスト変換ツールを用いて、構成ビットパターンに処理するとともに、Ｌチップ及びＸチップのネットリストファイルとしてのロジックチップにロードする。

【００３６】１.３.１.４　具体的なメモリモジュール設計

図２８は、好適例において用いられる、メモリモジュールの設計部を示す図である。これを、図２３に示した上述の説明に基づく構成に従って、アーキテクトすることに注意しなければならない。XC3090　ＬＣＡロジックチップに代わるＬチップソケットに、プラグを差し込み接続するように構成する。このようにして１２８個のＬ－Ｘパス、すなわち、各々３２個のＸチップにつながっている４個のパスを設ける。共通Ｉ／Ｏを有する３２Ｋ、８ビットスタティックＲＡＭチップを、８個のＲＡＭの各々の２個のバンク中に用いる。各バンクは、それ自体のＭＡチップ、XC2018　ＬＣＡを有している。各ＭＡチップは、８個のアドレスパス及び８個の書込みイネーブルを用いて、そのＲＡＭを制御する。各ＭＡチップを、モジュール中のすべてのＭＡ及びＭＤチップが共有している制御バスに接続するとともに、ホストインタフェースバスに接続する。残りのピンは、クロスバーと接続している。各々異なるＸチップとつながっている、２８個のＬ－Ｘパスを設ける。ＭＡチップ０は、一組のパス、パス０を使用し、ＭＡ１はパス１を使用する。これによ

って、２個の独立設計ＲＡＭに対する別々のアドレス及びコントロール回路網が与えられる。完全な３２個のＬ－Ｘパスよりも少ないパスを接続する。これは、単に、XC2018のピンの数が制限されているからに他ならない。設計変換の間、このモジュールにおけるミッシングパスに対応する、相互接続Ｌ－Ｘパステーブルにおけるパス構成要素が利用できないことに注意しなければならない。このため、回路網を、パス構成要素を介して相互接続できない。８個のＭＤチップには、すべてXC2018 ＬＣＡを使用する。３２個のＸチップを設ける場合、（上述の方法によれば）各々のＮＤチップは３２／８＝４個の異なるＸチップを接続している。各チップは、設計部メモリデータビットに用いられる２＊Ｍ／ｎ＝８個のパスを有している。その内の２個は、各Ｘチップにつながっている。各Ｘチップにつながっている付加的な２個のパスを設け、以下に示すように、モジュールを、１２８ビットベクトルメモリとして使用できるようにする。好適例において実現されるホストインタフェースバスを、Ｒバスと称する。Ｒバスは、すべてのＬチップポジションを、付加的なピンを用いて接続する。これについては、ホストインタフェースのセクションで説明する。５個の異なる設計部メモリ構成を、このモジュール中で用いることができる。以下のチャート及び図２８において“パス０”は、各Ｘチップからつながっている一組のＬ－Ｘパスを示しており、“パス１”は他の一組を示している。

・８ビット５１２Ｋの１個のメモリ：Ｌ－Ｘパス０及び１を介した（ＭＡ０及びＭＡ１の両方に接続できるように二重にしている）１９個のアドレス及び２個のコントロール（ＷＥ、ＯＥ）、Ｌ－Ｘパス２及び３を介した１６個のデータ（ＤＩ／ＤＯ又はドライバ／レシーバ）。各ＭＤチップは、１６個のＲＡＭに接続された１個のデータビットを有している。

・１６ビット２５６Ｋの１個のメモリ：Ｌ－Ｘパス０及び１を介した１８個のアドレス及び２個のコントロール、Ｌ－Ｘパス２及び３を介した３２個のデータ。各ＭＤチップは、各々８個のＲＡＭに接続されている２個のデータビットを具えている。

・３２ビット１２８Ｋの１個のメモリ：Ｌ－Ｘパス０及び１を介した１７個のアドレス及び２個のコントロール、Ｌ－Ｘパス２及び３を介した６４個のデータ。各ＭＤチップは、各々４個のＲＡＭに接続されている４個のデータビットを有している。

・８ビット２５６Ｋの２個のメモリ：各々、Ｌ－Ｘパスを介した１８個のアドレスと、２個のコントロールとを有している。パス０は一方のメモリ（ＭＡ０）のためのものであり、パス１は他方のメモリ（ＭＡ１）のためのものである。各々は、パス２及び３を介した１６個のデータを有している。各ＭＤチップは、８個のＲＡＭに接続された、各々のメモリのための１個のデータビットを有している。

・１６ビット１２８Ｋの２個のメモリ：各々は、Ｌ－Ｘパスを介した１７個のアドレスと２個のコントロールとを有している。パス０は一方のメモリのためのものであり、パス１は他方のメモリのためのものである。各々、パス２及び３を介した３２個のデータを有している。各ＭＤチップは、４個のＲＡＭに接続された各メモリのための２個のデータビットを有している。コントロールバスは、一般的にすべてのＭＡ及びＭＤチップに接続された１２個のパスから成っている。１２個のパスは最大コントロール構成を保持する必要がある。この構成は三つのアドレスビットである。すなわち、設計書込みイネーブルと、２個の２５６Ｋ、８ビット設計部メモリの各々のための設計部出力イネーブル信号とに、バス書込みイネーブル及びバス読出しイネーブルを加えたものである。

【００３７】１.３.２　刺激及び応答

リアライザシステムを多数使用することは、ホストコンピュータの刺激信号送信と、設計部への応答信号及び設計部からの応答信号の捕捉とに依存している。このことを、バッチ形式で行う場合、すなわち、信号の大部分を一度に送信及び収集する場合に、ベクトルメモリを用いる。このことを、一回に一つの信号で行う場合には、スティミュレータ及びサンプラを用いる。

【００３８】１.３.２.１　刺激を与えるためのベクトルメモリ

連続的且つ反復的な刺激のストリームを、シミュレーション適用のような、テストベクトルの高スピード反復適用のために実現される設計部中の一組の回路網に供給することが時々必要となる。このことは、実現される設計部の回路網にメモリをインタフェースさせること、刺激ベクトルをホストコンピュータからメモリに書込むこと、更には、順次にメモリを１回ないし数回読出し、刺激を設計部に送ることによって行われる。連続的且つ、リニアなメモリロケーションを読出す必要があるため、アドレスストリームを２進カウンタによって設ける。図２９はこのような刺激ベクトルメモリを達成するための手段を示している。規則的なクロック信号ＥＣＬＫはプロセスを制御する。ＥＣＬＫを周期化、すなわち、各刺激ベクトルの度毎にハイとローとを発生させる。２進カウンタはアドレスシーケンスを提供する。ＥＣＬＫがハイになると、カウンタは次の刺激ベクトルのアドレスまでカウントアップする。次の刺激ベクトルのアドレスは、ＥＣＬＫの周期の間ＲＡＭによって読出される。ＥＣＬＫが次にハイになると、ちょうど読出された刺激ベクトルの値がＤフリップフロップのクロックとなる。フリップフロップの出力信号は刺激ベクトルの値で刺激される回路を駆動する。フリップフロップは、ベクトル間に必要なクリーントランジッションを与える。その理由は、ＲＡＭ出力が正しい値に安定する以前に、その読出

サイクルの間変動し得るからである。このプロセスは繰り返され、一連の刺激ベクトルが実現される設計部に与えられる。この構造は繰り返され、刺激が多くの回路網に提供される。刺激ベクトルをＲＡＭに書込むのに用いられるホストコンピュータへのインタフェースは、簡単のため図示していないが以下に引用する図面により更に詳細に示す。

【００３９】１．３．２．２　応答捕捉のためのベクトルメモリ

同様に、実現される設計部からの応答を捕捉する一モードでは、連続的なサンプルのストリームすなわち一組の回路網からのベクトルを捕捉する。この時、ロジックアナライザが現実のハードウエアデバイスから捕捉を行う。このことは、メモリを、実現される設計部の回路網にインタフェースさせ、実現される設計部が順次に動作するときに回路網からのベクトルをメモリに書込み、更に捕捉された応答ベクトルを、解析のためにホストコンピュータへ戻すことによって行われる。連続的且つ、リニアな一連のメモリロケーションを読出す必要があるため、前記と同様、アドレスストリームを２進カウンタによって設ける。図３０はこのような応答ベクトルメモリを開発する手段を示している。刺激メカニズムのように、クロック信号ＥＣＬＫがプロセスを制御する。各応答ベクトルの度毎に、ＥＣＬＫの同期をとる。２進カウンタはアドレスシーケンスを提供する。ＥＣＬＫがハイになると、カウンタは次のベクトルのアドレスまでカウントアップする。ＥＣＬＫがローになると、応答ベクトルの値がトライステートドライバによってＲＡＭＤＱデータピンに伝達され、書込みのためにＲＡＭがイネーブルされる。ＥＣＬＫが再びハイになるとこの値はＲＡＭロケーションに書込まれ、ＲＡＭ書込みイネーブル及びトライステートドライバイネーブルはディゼーブルされ、カウンタは次のベクトルのアドレスまで進む。このプロセスは繰り返され、実現される設計部からの一連の応答ベクトルを記録する。この構造は繰り返され、刺激が多くの回路網に供給される。刺激ベクトルをＲＡＭに書込むために用いられるホストコンピュータへのインタフェースは、簡単のため図示していないが、以下で引用する図面において更に詳細に説明する。一般的に、実現される設計部を刺激し、これらの応答を発生させる。刺激が刺激ベクトルメモリから生じる場合、両ベクトルメモリは同一のＥＣＬＫ信号を用いている。ＥＣＬＫ信号は、新しいアドレスがカウンタから読み取られ、ＲＡＭをアドレス指定するとともに、データが読出され、刺激Ｄフリップフロップをセットアップするのに十分長くハイである必要がある。また、ＥＣＬＫ信号は、刺激が実現される設計部に影響を及ぼし、この影響に対するすべての応答が安定し、且つ、これらの応答がＲＡＭに書込まれるのに十分長くローでなければならない。刺激がいずれかから生じる場合、応答回路網を正しくサンプリングするために、応答ベクトルメモリのＥＣＬＫ信号は実現される設計部と同期されている必要がある。

【００４０】１．３．２．３　刺激及び応答のためのベクトルメモリ

図３１で示されているように、刺激及び応答ベクトルメモリシステムに関して上記のように定義された、刺激及び応答ベクトルメモリの機能を組合わせることができる。ＲＡＭビットは、たとえ同一のＲＡＭデバイスであっても、刺激又は応答のいずれか一方に自由に割当てることができる。その理由は、ＥＣＬＫがハイのときに刺激読出し機能が生じ、そして、ＥＣＬＫがローのときに応答書込み機能がこれに続くからである。トライステート応答ドライバを両方とも刺激Ｄフリップフロップ入力とし、同一のＲＡＭＤＱデータピンに接続することによって、一つのビットを刺激及び読出しの両方に用いることができる。シンプル刺激ベクトルメモリと組合せ刺激／応答ベクトルメモリとの重要な相違点は、刺激ベクトルを、１回だけＲＡＭから読出すことができるということである。その理由は、ＲＡＭビットを刺激のみに用いた場合でさえ、各メモリロケーションをＥＣＬＫの半周期のローの時に書込むからである。このことは、ＲＡＭチップのすべてのビットを刺激に用い、且つＥＣＬＫが書込みイネーブルを主張しない場合にのみ避けることができる。前の図面は、一般的な方法でベクトルメモリを実現したものを図示している。更に、点線は、いかにしてロジックチップ（“ＭＡチップ”及び“ＭＤ‘ｎ’”）を構成することでベクトルメモリロジック機能を実現することができるかを示すものである。これらロジックチップは、適切にＲＡＭチップ及びリアライザ相互接続（Ｘチップ）に接続されている。ベクトルメモリと、ソフトウエアからの刺激を電気的な型に再び戻す変換については、米国特許第４，７４４，０８４号明細書において詳細に説明されている。この内容を、参考のためにここで用いる。

【００４１】１．３．２．４　フォールトシミュレーションのためのベクトルメモリ

リアライザフォールトシミュレーションシステムについては、これについてのセクションにおいて説明する。フォールトシミュレーションでは、応答はベクトルメモリに捕捉されず、その代わりに、フォールト応答ベクトルメモリによって所定の良好な回路の応答と比較される。フォールト応答ベクトルメモリは、以下の点において上で示した簡易刺激ベクトルメモリと同一のものである。すなわち、ＭＤチップのフリップフロップの出力を用いて回路網を駆動する代わりに、出力はＸＯＲゲートによって回路網の値と比較される。ＸＯＲゲートを、ＥＣＬＫが同期をとるセットフリップフロップに接続し、回路網とメモリとの差を表示しているＸＯＲゲートがハイの場合フリップフロップをセットする。ホストは、ホストインタフェースを介してこのセットフリップフロップを

読出すことができ、差が検出されているかどうかを調べることができる。

【0042】1.3.2.5　実現される設計部におけるベクトルメモリの相互接続

実現される設計部へのベクトルメモリの接続方法は多くの方法が考えられる。1以上のロジックチップに直接接続され、及び／又は相互接続パスのいずれか又はすべてに接続されたベクトルメモリを用いて、リアライザシステムを設計することができる。例えば、ベクトルメモリを、Lチップ及びXチップを用いてロジックボードに取り付けることができるとともに、ボードとは離れているX－Yパスに接続することができる。ベクトルメモリを、Yレベルクロスバーのチップボードに取り付けるとともに、X－Y及びY－Zパスに接続することもできる。ベクトルメモリを、ロジックチップの代わりにLチップロケーションに取り付け、Lチップロケーションに作用するL－Xパスに接続するというテクニックもある。この場合、これらL－XパスをベクトルメモリとXチップとの間にのみ接続する。Xチップを構成することによって、実現される設計部の回路網への接続を行い、ベクトルメモリを回路網に接続する。この際。回路網はXレベル相互接続を介してつながっている。モジュールの方法でロジックチップをベクトルメモリモジュールに置き換え、リアライザシステムを、必要な数の又は必要よりも少数のベクトルメモリを用いて構成することができる。リアライザ設計部メモリを、Lチップロケーション中の1以上のロジックチップに代えて取り付けているため、このテクニックを用いて、共通ハードウエアメモリモジュールを設計部メモリモジュール又はベクトルメモリモジュールとして用いることができる。メモリモジュール中にロジックチップを構成するとともに、リアライザシステムの相互接続を適切に行うことによって機能を選択する。これは、好適例において用いられているベクトルメモリアーキテクチャである。

【0043】1.3.2.6　特別なベクトルメモリ設計部

好適例において、共通メモリモジュールを、設計部メモリ及びベクトルメモリ応用の両方のために使用する。その一般的なアーキテクチャ及び設計は、設計部メモリのセクションにおいて説明し、ここでは説明しない。いかにして、モジュールをベクトルメモリとして用いるかの詳細については、以下に示すとおりである。以下の2個の図面は、ホストインタフェースからの完全読出し／書込みアクセスを用いて、組合せ刺激／応答ベクトルメモリのためのMA及びMDチップ中に前記と同様のロジックを構成することを示している。ホストコンピュータが非活動状態である場合、すべての動作は上記簡単な例にて示したのと同一のテクニックに従っている。図32において、ホストインタフェースを介してホストが出力するECLK信号を、相互接続を介してMAチップに相互接続している。ECLK信号は、各MAチップで構成されるアドレスカウンタの同期をとる。各々、一組のRAMを制御している1以上のMAチップをモジュール中に設けているので、各MAチップは、ベクトルアドレスカウンタのコピーを有している。すべてのカウンタは、同一のコントロール（ECLK及びバスインタフェースからのリセット信号）を得ているため、その各々は常に他のカウンタと同一のアドレスを送信する。通常（バスインタフェースが非活動状態の場合）、アドレスがカウンタ出力から送られ、RAMのアドレス指定を行う。ECLKがロー状態（書込み応答位相）の場合、デコーダロジックは、前述の例と同様にすべてのRAM書込みイネーブルを主張する。ECLKは、コントロールバスにも伝達され、MDチップのロジックを駆動する。MDロジックは刺激及び応答ベクトル値それ自体を処理する（図33）。通常（バスインタフェースが非活動状態の場合）、ECLKがハイ状態のとき、RAMは刺激ベクトル値を読出し、ECLKがロー状態になると、RAMとフリップフロップとを同期させる。フリップフロップは、上記と同様に各回路網に刺激を与えるためのものである（その内の一つを図示する）。従って、刺激を相互接続Xチップを介して回路網へ伝達する。ECLKがロー状態の場合、すべてのトライステートイネーブル（e0、e1、・・・en）が主張され、相互接続（2個を図示する）を介して回路網から出力される応答値をマルチプレクサを介してRAMDQデータピンに伝達する。ホストコンピュータが、ホストインタフェースバス（特に、好適例のRバス）を介してこのメモリをアクセスする場合、各々のMAチップ中に構成されるバスインタフェースロジックが活動状態となる。これは、アドレスマルチプレクサ（mux）を切り換え、バスがRAMのアドレス指定を行う。バスサイクルがRAMに書込みを行うためのものである場合、デコーダロジックは、アドレスビットを用いてどのRAMに書込みを行うべきであるかを解読するとともに、適切な書込みイネーブル信号を出力する。RAMを選択するのに必要とされるアドレスビット及び読出し及び書込み制御信号も、コントロールバスを介してMDチップに伝達される。MDチップにおいては、バスが読出しサイクルを行う場合、デコーダロジックはすべてのトライステートRAMDQピンドライバをディゼーブルし、アドレスビットを用いて読出しマルチプレクサを介してアドレス指定されたRAMのDQデータ出力を選択し、更には、バス読出しイネーブル信号がデータ値をこのビットのためのホストインタフェースバスのデータラインに伝達する。バス書込みサイクルにおいて、デコードロジックは、書込みマルチプレクサを用いて、応答を与える回路網ではなく、ホストインタフェースバスのデータラインから生じるデータ値を選択するとともに、アドレス指定されたRAMのためのトライステートRAMDQドライバをイネーブルし、データ

をＲＡＭ入力へ伝達する。

【００４４】１.３.２.７　ベクトルメモリの設計変換及び仕様

回路網をベクトルメモリに接続すべきであるということを説明するために、ユーザは、回路網に入力設計に関する特別な特徴を付加し、特定のベクトルメモリ及び接続が刺激のためのものであるのか又は応答のためのものであるのかを説明する。設計部変換方法は、一組の所定の部分的ネットリストファイルに基づくものであり、この内の一つは各モジュールのロジックチップのためのものであり、前記と同様ベクトルメモリ刺激及び応答接続と、ベクトルメモリデータパス及びコントロールロジックと、バスインタフェースロジックとに関するステートメントを用いている。この方法では、ＥＲＣＧＡネットリスト変換ツールは、任意の出力端子又はＩ／Ｏピンに接続されていない入力端子、及び任意の入力端子又はＩ／Ｏピンに接続されていない出力端子のような、通常は接続されていないネットリストファイル中の基本要素及び回路網のためのロジック及び相互接続を構成することはない。各ベクトルメモリビットに対する刺激接続及び応答接続のためにロジックを設ける。ネットリストに供給されるいずれか一方の相互接続のみが実際に構成され、他方は構成されない。その理由は、通常それをネットリストに接続しないからである。予め定義されたファイルは、相互接続を用いてベクトルメモリ刺激接続とベクトルメモリ応答接続とを接続するのに用いるモジュールＩ／ＯピンのＩ／Ｏピン数の仕様を除いて、完全なものである。各ファイルにおける刺激及び応答接続の数を、何個のＩ／Ｏピンをファイルのロジックチップ中に用いることができるか、どの程度のロジックを各チップに、更には全体としてどの程度のロジックをモジュールに設けることができるか、によって決定する。その方法は、以下のとおりである。：設計部変換のセクションにおいて説明したように、以下のようなベクトルメモリの特別な例外を有するものの、一般的な方法を設計部変換に用いる：

・設計部リーダは、ベクトルメモリ接続のために設けられた回路網を識別するために入力設計ファイルからの特性情報を読出し、且つ、バスインタフェースロジックではなく、回路網に接続された１以上のベクトルメモリ基本要素を、その設計部データ構造に組込む。設計部リーダは、ホストインタフェースクロック発生器及びベクトルメモリ基本要素に接続されたＥＣＬＫ回路網を作り出す。

・パーティショナとは、ユーザが、メモリモジュールを取り付けるボード上のいずれかのチップ指定するということである。このデータに基づき、パーティショナは、ベクトルメモリ基本要素を通常の方法でメモリモジュール中に分割する。

・インタコネクタは、他のロジックチップ基本要素と同一のベクトルメモリ基本要素を処理し、これらを回路網中の他の基本要素を用いて接続しているＬ－Ｘパスを決定する。

・リアライザシステム中の各ロジックチップのネットリストファイルに書込みを行う場合、各ベクトルメモリ回路網接続は以下によってネットリストされる：

１）どのロジックチップが、相互接続手続によって基本要素が選択したパスを接続するかを決定する。

２）通常のロジックチップＩ／Ｏピンナンバを得る際に説明したのと同様の手続を用いて、パスナンバ及びロジックチップナンバからロジックチップＩ／Ｏピンナンバを得る。

３）これまで他の回路網に割当てられていないロジックチップに関する回路網から、予め定義された刺激又は応答ベクトルメモリ接続を選択する。

４）ステートメントをこのロジックチップのネットリストファイルに加え、このロジックチップＩ／Ｏピンナンバを、予め定義されたベクトルメモリ接続に接続するために用いることを明示する。

・設計変換システムは、対応テーブルファイルも送出し、回路網の名称をベクトルメモリ及びベクトルメモリビット位置と関連づけ、動作中使用する。

・ＥＲＣＧＡネットリスト変換ツールは、用いられるベクトルメモリ刺激及び応答入力端子のロジック及び相互接続のみを構成する。

【００４５】１.３.２.８　スティミュレータ

スティミュレータは、単一の記憶ビットとし、ホストコンピュータで制御し、設計部の回路網を駆動する。スティミュレータは、ホストが入力信号を設計部に供給するのに用いられる。２種類のスティミュレータ、すなわち、ランダムアクセスタイプとエッジ検知タイプとを設ける。実際のランダムアクセススティミュレータは、フリップフロップであり、その出力信号はホストインタフェースバスを介し、ホストが必要に応じてデータをロードする設計部回路網を駆動する。ランダムアクセススティミュレータは、設計部の動作を変化させることなく、他の刺激された回路網に呼応して、常に値を変化させることのできる回路網を刺激するのに用いられる。このような回路網の一例としてはレジスタへのデータ入力がある。各スティミュレータは唯一のバスアドレスを有し、ホストがデータをこのアドレスに書込む場合に、バスインタフェースロジックは、データをＤ入力に与えるとともにスティミュレータフリップフロップのクロック入力の同期をとる（図３４）。エッジ検知タイプのスティミュレータは、設計部の動作、例えばレジスタへのクロック入力を修正するための他の回路網と同期しながら変化しなければならない回路網を刺激するのに用いられる。第２フリップフロップを、ランダムアクセススティミュレータと設計部回路網との間に配置する。同期をとらなければならない一群のこのようなスティミュレータのす

べてを共通クロックに接続する。新しい一組の回路値を入力するために、ホストは、新しい値を、たとえどのようなオーダであっても、上記と同様に、ホストインタフェースバスを介して各スティミュレータの第１フリップフロップにロードする。新しい値が設計部にすべて供給される必要がある場合、ホストは、共通‘同期クロック’を周期化し、一度にすべての値を第２フリップフロップにロードし、このようにしてすべての回路網を同時に駆動する（図３５）。

【００４６】１.３.２.９　サンプラ

サンプラは、単一の記憶ビットであり、ホストコンピュータによって制御され、設計部の回路網を受信する。サンプラはホストによって使用され、設計部からの出力信号を捕捉する。サンプラの最も簡単な形は、Ｄ入力端子で設計部回路網を受信し、同期をとることができ、且つ、ホストインタフェースバス及びバスインタフェースロジックを介してホストが必要に応じて読出すことのできるフリップフロップである。通常、多数のサンプラを、共通‘サンプルクロック’に接続する。サンプラデータ出力は、‘サンプルクロック’出力と同様に、唯一のバスアドレスを有している。ホストは、クロックを周期化し、一群のサンプルを取り出し、その後、サンプリングされたデータ値を一つ一つ読出す（図３６）。必要とされるホストＩ／Ｏの数を削減するために、第２フリップフロップを付加的に加え、変化検出サンプラを構成する。第２フリップフロップを、サンプリングフリップフロップと同一のクロックに接続し、その入力端子をサンプラの出力端子に接続する。結果的に、第２フリップフロップは最も新しいクロック周期前にサンプラが有していた値を保持している。２個のフリップフロップ出力をＸＯＲゲートで比較する。ＸＯＲゲートは、サンプリングされた値の変化のため２個のフリップフロップが相違する場合にハイ状態の値を出力する。一群のサンプラからの全ＸＯＲ出力信号をホストが読出し可能なＯＲゲートによって加算する。上述したように、‘サンプルクロック’を周期化することによって、回路網をサンプリングした後、ホストは、まず第１にこのＯＲゲートの‘変化’値をチェックし、グループ中のどの値が変化したのかを調べる。変化していない場合には、これらサンプラのいかなる値をも読出す必要がない（図３７）。

【００４７】１．３．２．１０　スティミュレータ及びサンプラの設計変換及び仕様

サンプラ及びスティミュレータフリップフロップ、ロジックゲート及びバスインタフェースロジックを、リアライザシステムロジックチップ中に実現する。回路網を、サンプラ又はスティミュレータに接続すべきであることを説明するために、ユーザは、回路網に、入力設計に関しての特別な特性を与え、スティミュレータ又はサンプラの特定のタイプとグループの同一性とを識別する。スティミュレータ及びサンプラを構成し、これらを、設計部の残りの部分及びバスインタフェースに接続するために、設計変換ソフトウエアシステムを用いる一般的な方法は以下に示すとおりである：設計変換のセクションにおいて説明したように、以下のようなスティミュレータ及びサンプラに関しての特別な例外があるものの、一般的な方法を設計部変換に用いる：

・設計部リーダは、特性情報を入力設計ファイルから読出し、スティミュレータ及び／又はサンプラのために設けられた回路網を識別し、バスインタフェースロジックではなく、回路網に接続されたスティミュレータ及びサンプラの基本要素を設計データ構造に組み込む。

・システムパーティショナは、このような基本要素の各々が、ロジックチップ中に何個の等価ゲートを有しているかについてのデータベースを有している。システムパーティショナは、バスインタフェースロジックの等価ゲート指数も有している。このデータに基づき、システムパーティショナは、その通常の分割化アルゴリズムに従ってスティミュレータ及びサンプラをロジックチップに割当てる。この際、システムパーティショナが、バスインタフェースロジックのサイズによって、ロジック容量の限界を小さくするという付加的な条件を課し、１以上のスティミュレータ及び／又はサンプラを有するロジックチップの各々が、バスインタフェースロジックブロックを有しなければならないということを説明する。

・インタコネクタは、他の基本要素と同じく、スティミュレータ及びサンプラ基本要素を処理する。

・リアライザシステムにおける各ロジックチップのネットリストファイルに書込みを行う場合、以下の手続を用いて、各サンプラ又はスティミュレータ基本要素をネットリストする。

１）サンプラ又はスティミュレータを構成するゲート及び／又はフリップフロップの基本的ステートメントを、ステートメント分割化のためのロジックチップのネットリストファイルへ送信する。相互接続基本要素について説明したのと同様の方法に従って、サンプリング又は刺激される回路網に亘る付加的な回路網のネームを、サンプリング又は刺激される回路網のネームから得る。

２）これが、この特別なロジックチップファイルにネットリストされる第１スティミュレータ及びサンプラである場合、バスインタフェースの予め定義されたネットリストファイルセグメントを用い、バスインタフェースを構成する基本要素及び回路網をロジックチップに供給する。インタフェース毎に１回のみ使用されるバスインタフェース相互接続には前記ファイルセグメントで定義される標準的なネームが与えられる。スティミュレータ又はサンプラロジックに接続されるものには、捕捉した回路網のネームが与えられる。このネームは、ステップ１において、基本要素を出力する際に用いたネームと整合している。

簡単ではあるが、一般的ではない方法を用いて、メモリ

モジュール又はユーザ指定のデバイスモジュールのロジックチップ中にのみ、スティミュレータ及びサンプラを実現する。このことは、ＥＲＣＧＡネットリスト変換ツールがどの出力端子又はＩ／Ｏピンにも接続されていない入力端子、及びどの入力端子又はＩ／Ｏピンにも接続されていない出力端子のような、通常接続されていないネットリストファイル中の基本要素及び回路網のためのロジック及び相互接続を構成しないということを仮定している。このことは、一組の予め定義された部分的ネットリストファイルに基づくものである。このファイルの一つは、各モジュールのロジックチップのためのものである。この際、以下のステートメントを用いている。

１）すべて共通‘同期クロック’に接続されている多数のエッジ検知タイプのスティミュレータ。

２）すべて同一の‘サンプルクロック’に接続された多数の変化検出サンプラ。

３）上記のすべてのためのバスインタフェースロジック。

予め定義されたファイルは、サンプラとスティミュレータとを相互接続を用いて接続するのに用いられるモジュールＩ／ＯピンのＩ／ＯピンナンバのＩ／Ｏピンナンバの仕様を除き、完全なものである。コントロールバスを用いて、同期及びサンプルクロックのような共通信号をロジックチップ間に分配する。各ファイル中のスティミュレータ及びサンプラの数を、何個のＩ／Ｏピンがファイルのロジックチップ中に利用できるか、どの程度のロジックを各チップが有することできるか、及び全体としてのモジュールによって決定する。その方法は、以下に示すとおりである：設計変換のセクションにおいて説明したような、一般的な方法を設計変換に用いる。この際、スティミュレータ及びサンプラに関して、以下のような例外がある。：

・設計部リーダは、スティミュレータ及びサンプラのために設けられた回路網を識別するために、入力設計ファイルからの特性情報を読出し、且つ、バスインタフェースロジックではなく、回路網に接続されたスティミュレータ及びサンプラ基本要素を、その設計部のデータ構造に組込む。

・パーティショナとは、ユーザが、メモリモジュール及びユーザ指定のデバイスモジュールを取り付けるボード上のいずれかのＬチップを指定するということである。このデータに基づき、パーティショナはまずメモリ及びＵＳＤ基本要素をモジュールに割当て、その後、その通常の分割化アルゴリズムに従い、各モジュール単位で利用することのできる数の限界に至るまで、スティミュレータ及びサンプラ基本要素をこのようなモジュールの残りへと分割化する。

・インタコネクタは、他のロジックチップ基本要素と同一なスティミュレータ及びサンプラを処理し、これらを、回路網で他の基本要素を用いて接続しているＬ－Ｘパスを決定する。

・リアライザシステム中の各ロジックチップのネットリストファイルに書込みを行う場合、各サンプラ又はスティミュレータ基本要素は以下によってネットリストされる：

１）どのロジックチップが、相互接続手続によって基本要素が選択したパスを接続するかを決定する。

２）通常のロジックチップＩ／ＯピンナンバをＩ／Ｏピンナンバを得る際に説明したのと同様の手続を用いて、パスナンバ及びロジックチップナンバからロジックチップＩ／ＯピンナンバをＩ／Ｏピンナンバを得る。

３）これまで他の回路網に割当てられていないロジックチップに関する回路網から、予め定義されたスティミュレータ／サンプラを選択する。

４）ステートメントをこのロジックチップのネットリストファイルに加え、このロジックチップＩ／ＯピンナンバをＩ／Ｏピンナンバを予め定義されたサンプラ／スティミュレータに接続するために用いることを明示する。

・ＥＲＣＧＡネットリスト変換ツールは、使用するスティミュレータ、サンプラ及び関連あるバスインタフェースロジックのためのロジック及び相互接続を構成する。両方の方法において、設計部変換システムは対応テーブルファイルも出力し、動作中に使用するために、回路網ネームを特定のスティミュレータ及びサンプラに関連させるとともに、アドレスをホストインタフェースバスで通信する。

【００４８】１．３．３　ユーザ指定デバイス

構成されたロジック及び相互接続チップの形態で実際に動作するハードウエア中に入力設計部を実現するために、他の実際のハードウエアデバイスをリアライザシステムに接続することが実用的であり且つ望ましい。マイクロプロセッサ又は他のＶＬＳＩ　ＩＣチップ、デジタル／アナログコンバータ、ディスプレイデバイス、入力キーボード及びスイッチ、記憶デバイス、コンピュータ入力／出力バス等のデジタル入出力を具える任意のデバイスを設けることができる。これらを、回路ボード又は大規模スケール構成素子のような、実現される設計部の一部を構成するデジタルシステムの一部とすることができる。これらのデバイスは、リアライザシステムのロジックゲート、フリップフロップ及びメモリ中で具現化することのできない、実現されるべき入力設計部の一部を示している。これは、ディスプレイのような物理的な理由によるもので、大規模記憶デバイスのような、リアライザシステムのリソースが不足しているため、又は標準的マイクロプロセッサのように、ロジック的な記述を利用することができないためである。代わりに、これらのデバイスは、すでに構成され正しいことが証明されている半通例のゲートアレイチップのような、ユーザがリアライザシステムリソースを用いて実現することを望まないデバイスともなり得る。その理由は、これを実現するために、リアライザシステムリソースを用いる必要がな

いため、又はユーザが設計部の実現される部分のデバイスを用いての正確な動作を試験したいためである。これらのデバイスは、リアライザシステムの一部ではないが、ユーザの設計の必要に応じて、ユーザによって指定されるものであるため、これらのデバイスを“ユーザ指定デバイス”（ＵＳＤ）と称する。ユーザがこのようなデバイスをリアライザシステムハードウェアに接続するのに用いるような標準的な手段を、リアライザシステムに設けるのに役立つような、多様なＵＳＤを設ける。この手段はユーザ指定のデバイスモジュール（ＵＳＤＭ）である。

【００４９】１.３.３.１　ユーザ指定のデバイスモジュール

ユーザ指定のデバイスモジュールは：

１）物理的にユーザ指定のハードウエアデバイスを接続する手段を具えている。

２）ＵＳＤとリアライザシステムロジック及び／又は相互接続チップとの間を接続している。ＵＳＤが、ロジックチップと類似する設計部の役割を果たすため、ロジックチップと同様の方法で、ＵＳＤＭを相互接続するのが好都合である。

２）通常、Ｌチップロケーションに取り付けられたロジックチップが行うように、ＵＳＤピンを相互接続ピンに自由に割当てる機能を設ける。

ユーザ指定のデバイスモジュールは、メモリモジュールがそのＲＡＭチップに有しているのと類似の機能を具えている必要があるので、ＵＳＤＭのアーキテクチャは、メモリモジュールのアーキテクチャと類似している。図３８は、ＵＳＤＭアーキテクチャを示している。ＵＳＤＭプリント回路基板、すなわち、ＵＳＤＭにプラグで取り付けられている移動可能なドーターカード（daughter card）の領域である、ユーザ指定のデバイス取り付け領域、又はマイクロプロセッサ、エミュレータ（emulator）計器と共通する方法で、ケーブルを介して接続されている他のこのような領域にデバイスを取り付ける。端末のブロックは、デバイス入出力ピンと、ＵＳＤＭロジックチップとの間に、コネクタ端末細条、一組のプリント回路ボードパッド、又は他のこのような手段を介して電気的な接続を行うための手段を具えている。端末のブロックは、デバイスの電源も具えている。物理的な、端末ブロックピンの容量が許す限り、１以上のデバイスを取り付けることができる。その代わりに、デバイスを、一般的な方法で、ケーブル及び中継装置を介して、遠隔的に接続することもできる。ＭＡ及びＭＤロジックチップの各々は、端末ブロックに接続されたＩ／Ｏピンと、相互接続に接続されたＩ／Ｏピンとを具えている。これらのチップを、メモリモジュールアドレス及びデータバスロジックチップにおいて説明したのと同様の方法で相互接続に接続する。付加的に、図にて示したように、メモリモジュールにチップを使用するのと同様の目的のために、これらのチップをホストインタフェースバス及び／又は共通コントロールバスにも接続する。一般的に、バスデータビットがＭＤチップに分配され、これによって、相互接続に分配されるように、ＵＳＤアドレス及びデータバスをＭＤチップに接続する。ＭＡチップをＵＳＤコントロールライン及び付加的にＵＳＤアドレスラインに使用する。図面は、可能性を説明するために接続された３個の仮説的ユーザデバイスを示している。ＵＳＤ０は、ＭＤチップを介して接続されたデータ及びアドレスバスと、ＭＡ０を介して接続されたコントロールラインＡ、Ｂ及びＣとを有している。ＵＳＤ１は、ＭＤチップに接続された３個のデータバスと、ＭＡチップを介してのアドレス及びコントロール接続とを有している。ＵＳＤ２は、アドレス指定のためのＭＡ１と、データのためのＭＤチップとを使用する。任意の特定のケースにおいて、リアライザシステムのユーザは、これらの設計及び使用にとって適切な方法を用いてこれらのＵＳＤを接続することができる。前記セクションにおいて示したように、メモリモジュールＭＤチップにおいて、双方向性ＲＡＭＤＱピンを相互接続するのと同様の方法を用いて、双方向性ＵＳＤ接続を相互接続する。相違点は、入力設計部の回路網を、出力イネーブルコントロールのようにして明示する必要があるという要件である。この回路網を、メモリモジュール数が２５及び２６の場合に示される“設計出力イネーブル”と同様の方法で相互接続ロジックに接続し、ＭＤチップの双方向性ドライバを制御する。通常、適切な出力イネーブルコントロール回路網を入力設計部中に設けていない場合、ユーザはこれを設ける必要がある。

【００５０】１.３.３.２　好適例のＵＳＤＭ

図３９において示した好適例において、ＲＡＭチップの代わりにＵＳＤを取り付けるための領域に関して、ＵＳＤＭはリアライザメモリモジュールと同一である。８個のＭＤチップの各々は１６個までのＵＳＤピンを相互接続し、２個のＭＡチップの各々は２３個までのＵＳＤピンを相互接続する。図は、２個の実際に取り付けられたＶＬＳＩデバイス、すなわちモトローラＭＣ６８０２０の３２ビットマイクロプロセッサ（“MC68020 32 Bit Microprocessor User's Manual”, Motorola, Inc., Phoenix, 1984）と、モトローラＭＣ６８８８１浮動小数点コープロセッサ（“MC68881 Floating Point Coprocessor User's Manual”, Motorola, Inc, Phoenix, 1985）とを示している。これらのデバイスは、ＵＳＤの優れた例である。その理由は、一般的にこれらのデバイスをデジタルシステムの設計に用い、これらのロジック回路網表現をユーザが利用可能とすることはできないからである。これらのデバイスは、以下の入／出力ピンを有しており、この詳細については以下に示すとおりである。

ＭＣ６８０２０

データ　　：D31-D0、双方向性
出力イネーブル条件：R/Wが“書込み”を示し、且つDBENが真である場合、D31-D0は、出力信号を送信し、そうでない場合には、入力信号を受信する。
アドレス　：A31-A0、出力
中央入力端子：CLK、DSACK0、DSACK1、AVEC、CDIS、IPL0-IPL2、BR、BGACK、RESET、HALT、BERR
中央出力端子：R/W、IPEND、BG、DS、DBEN、AS、RMC、OCS、ECS、SIZ0、SIZ1、FC0-FC2
MC6888A
データ　　：D31-D0、双方向性
出力イネーブル条件：R/Wが“読出し”を示し、且つDSACK0及び/又はDSACK1が真の場合、D31-D0は、出力信号を送信し、そうでない場合には、入力信号を受信する。
アドレス　　：A4-A0、入力
中央入力端子　　：CLK、SIZE、RESET、AS、R/W、DS、CS
中央出力端子　　：DSACK0、DSACK1
データバスとアドレスバスとをMDチップを用いて相互接続する。メモリデータバスのセクションにおいて説明したように、バスデータビットをクロスバーを横切ってスライスし、図に示すように相互接続を容易にしている。制御信号はMAチップによって相互接続される。出力イネーブル制御信号は、上述したように、制御信号に接続された特別のロジックによって発生される。ユーザは、このロジックを入力設計部に設け、設計部の残りの部分を用いてLチップ中に実現する。各MDチップが異なるL-Xバスの組を接続し、通常出力イネーブルコントロールが全バスに関して一般的なものであることから、設計部変換システムは、これらの回路網をMAチップの内のいずれか1個に接続するとともに、USDMコントロールバスを用いて、回路網を接続の必要性があるMD及びMAチップに接続するように、MA及びMDチップを構成する。

【0051】1.3.3.3　ユーザ指定のデバイスのための設計部の変換

USDを特別の基本要素を用いて入力設計部内に設ける。USDは、ユーザが作成するUSD仕様ファイルを示している特性データを伝達する。このファイルは、どのLチップロケーションにこのデバイスを有するUSDMを取り付けるかを示すとともに、USDのI/Oピンをリストする。この際、入力設計部のUSD基本要素中に使用されているピンネームを使用している。各ピンに対して、USDは、ピンを接続しているUSDMロジックチップ及びピン数と、ピンが入力、出力又は双方向性であることとをリストする。ピンが双方向性である場合には、入力設計部中の出力イネーブルコントロール回路網のネームもリストする。設計部変換ソフトウエアシステムは、USDを構成するとともに、USDを設計部の残りの部分に接続するネットリストファイルを出力する。一般的な方法を使用するが、これには、以下のようなUSDに対する例外がある：

・設計部リーダは、USD基本要素を設計部データ構造中に読出す。設計部リーダは、USD仕様ファイル中で読出しを行うために、ファイル特性を使用するとともに、後の使用のための基本要素記憶と関連する情報を記憶する。基本要素記憶を、各々異なる出力イネーブルコントロール回路網に接続された特別のピンに供給する。
・変換ステージは、構成が利用可能であり、且つ、ピンが正しく構成と対応していることをチェックする。
・システムパーティショナは、USDをUSD仕様ファイルで特定されるLチップロケーションに配置する。
・インタコネクタは、USDピンに接続された回路網を特定のL-X相互接続バスに割当てる。インタコネクタはこれを行う際、USDピンに接続された回路網を、USD仕様ファイルで特定されるMA又はMDチップに接続するパスにのみ割当てることができ、且つ、イネーブルコントロール回路網ピンを、MAチップに接続しているパスにのみ割当てることができるという制約を条件としている。
・ネットリストファイルをUSDMに送信するために：このUSDMのUSDを制御している各々の出力イネーブルコントロール回路網が：基本要素を、この回路網のMAチップのネットリストファイルに送信する：その理由は、この回路網のために使用するL-Xバスを受信している入力バッファが、出力バッファの入力を駆動し、これによって、この回路網に割当てられたコントロールバスラインを駆動するからである。このUSDMのUSDに接続されている各回路網が：USD入力ピンを駆動する場合、基本要素を、このピンのロジックチップのネットリストファイルに送出する：その理由は、この回路網に使用される受信バスからの入力バッファが、このUSDピンのために用いられる端末ブロックピンを駆動する出力バッファの入力端子を駆動するからである。USD出力ピンを受信する場合、基本要素を、このピンのロジックチップのネットリストファイルに送信する：その理由は、この回路網に用いられる駆動バスにつながっている出力バッファが、このUSDピンに用いられる端末ブロックピンを受信している入力バッファの出力を受信する。USD双方向性ピンに接続している場合、基本要素を、このピンのロジックチップのネットリストファイルに送信する：その理由は、この回路網に用いられている受信バスからの入力バッファが、このUSDピンに用いられる端末ブロックピンを駆動しているトライステート出力バッファのデータ入力端子を駆動し、この回路網に用いられている駆動バスにつながっている出力バッファが、このUSDピンに用いられている端末ブロックピ

ンを受信する入力バッファが一方の入力端子を駆動する２入力ＡＮＤゲートの出力を受信し、このピンの出力イネーブルコントロール回路網に割当てられたコントロールバスラインからつながっている入力バッファが、トライステート出力バッファのイネーブル入力端子及びＡＮＤゲートの他の入力端子を駆動するからである。

【００５２】1.4　構成

ロジック及び相互接続チップ技術のセクションにおいて説明したように、各チップの構成ビットパターンが、ＥＲＣＧＡネットリスト変換ツールによって出力される。リアライザ設計変換システムの最終ステージが、すべてのチップの発生する構成ファイルから設計部の単一バイナリー構成ファイルへと送られるデータを捕捉する。これによって、データがホストコンピュータ中に永久に記録される。リアライザシステムを各々使用する前に、構成ファイルからデータを読出し、このデータをホストインタフェースを介してリアライザハードウエアに伝達し、更にチップにロードすることによって、使用する設計部のロジックチップ及び相互接続チップを構成する。ホストインタフェースと、システム中のすべてのロジックチップ及び相互接続チップとの間に、構成接続を設ける。チップを構成すると、すべてのロジック機能及び相互接続の合計は入力設計部によって特定されるものと一致し、設計部が動作できる。好適例においては、Xilinx製のＬＣＡをロジックチップ及びクロスバーチップとして用いる。バイナリー構成ビットパターンを、１回に１ビットずつ、ＬＣＡ構成メモリのシリアルシフトレジスタにロードすることによってＬＣＡを構成する。各ビットを構成データ入力端子（ＤＩＮ）に供給するとともに、１回構成クロック（ＣＣＬＫ）を周期化してロードする。各ＬＣＡとホストインタフェースとの間の特別な構成接続は設けていない。その理由は、システムが全部で３５２０個までのロジックチップ及びクロスバーチップを具えなければならないからである。その代わりに、マルチビットデータバスと構成クロックとを有する構成バスを設け、これを、ＬＣＡを有するすべてのボードに接続する。構成を行うために、ループ毎に、データバス中のビット数と同数のチップを用いてロジックチップ及びクロスバーチップをグループ化する。１グループ中のすべてのチップを並列に構成する。図４０に示しているように、グループ中の各々のＬＣＡは、バスデータバスの異なるビットに接続された構成データ入力端子を有している。各グループにおける構成コントロールロジックブロックを、ホストインタフェースバスと、バス構成クロックと、グループ中のすべてのＬＣＡのクロック入力端子とに接続する。これらのコントロールロジックブロックを、ホストインタフェースバスを介して、ホストの命令によって選択的にイネーブルし、ホストインタフェースバスが意図するＬＣＡのグループがクロック信号を受信できるようにし、このような構成としている。これは、ホストコンピュータがリアライザシステムを構成するのに行う手続である。制御及びデータ伝送はすべてホストインタフェースを介して行われる：すべてのロジックチップ及びクロスバーチップを構成するために：各構成グループは：このグループのコントロールロジックブロックが、構成クロックをチップに送るように指示する。１個のＬＣＡ中の構成ビットと同数の周期の間：このグループ中の各チップの１構成ビットをバスデータバスにロードする。バス構成クロックを１回周期化する。次の周期へ。このグループのコントロールロジックが、もはや構成クロックを送信しないように指定する。次のグループへ。

【００５３】1.5　ホストインタフェース

ホストコンピュータの制御の下、リアライザシステムは周辺装置として動作する。ホストコンピュータは、設計部の構成ファイル中に記憶された構成ビットパターンを用いて、設計部に従ってリアライザシステムのロジックチップ及び相互接続チップを構成する。ホストコンピュータは、その外部リセット及びクロック信号を制御することによって連続的な設計部の動作を制御する。従って、ホストコンピュータは、スティミュレータ、サンプラ及びベクトルメモリを制御するとともに、ベクトル及び設計部メモリの内容を読出し及び書込むことにより、設計部と相互的に作用する。ホストコンピュータは、これらすべてをリアライザシステムホストインタフェースを介して行う。ホストコンピュータは、リアライザシステムのホストインタフェース及び構成バスを制御する。

【００５４】1.5.1　ホストインタフェースアーキテクチャ

リアライザシステムホストインタフェースを、完全に慣用となっているラインに沿って構成する（図４１）。リアライザシステムは、ホストインタフェースバスコントローラ、構成バスコントローラ、クロック発生器及びリセットコントローラを具えている。これらの各々について以下に説明する。インタフェースを、リアライザハードウエアシャーシのボードに構成するとともに、ケーブル及びインタフェースカードを介して、ホストコンピュータのＩ／Ｏバスに接続する。ホストインタフェースのコントロール機能を、特定のコンピュータの要求に応じて、ホストコンピュータのメモリアドレススペース、又は、入力−出力バススペースのいずれかに作成する。

【００５５】1.5.2　ホストインタフェースバス

ホストインタフェースバスを、リアライザシステム中の、正規のロジックチップ及びメモリモジュールロジックチップの幾つか又はすべてのＩ／Ｏピンに接続する。ホストインタフェースバスは、リアライザシステムコントロール及びデータアクセス機能を割当てるアドレススペースを具えている。ホストは、最適なバスマスタであり、ホストインタフェースバスコントローラを介して、アドレス指定された読出しコマンド及び書込みコマンド

をバスに送出する。ホストは、データをリアライザシステム機能とホストとの間に伝送する。ホストインタフェースコントロールロジックブロックを、メインロジックチップ及びメモリモジュールロジックチップにプログラムし、リアライザシステム機能を、バスを介して制御できるようにする。このバスによって制御される機能の特別な例としては、サンプラ、スティミュレータ、ベクトルメモリアドレス指定、オペレーション、ホストデータアクセス、及び設計部メモリホストデータアクセスがある。これらのコントロールブロックは、ロジックチップにすべてプログラムされているため、バスアドレススペース中の特別な機能及びロケーションを、すべてロジックチップのプログラミングによって規定し、任意の所定の設計部又は動作モードの必要に応じて変化させることができる。ホストインタフェースバスの特定の設計部は、特定のリアライザシステムを具体化した場合のデータアクセススピード及びハードウエアピンの利用可能性に依存している。好適例では、Rバスと称する１１ピンホストインタフェースバスを、すべてのロジックチップの専用Ｉ／Ｏピンに接続している。好適例のハードウエアは、データ及びアドレスのために用いられる８個の双方向性ラインと、クロックと、２個のコントロールラインとを具えている。Rバスは３２ビットアドレススペースと８ビットデータ幅とを有しており、ホストが４０億までのユニークロケーションから８ビットデータを読出し又は書込みできるようにしている。Rバスを、アドレスレジスタ、データレジスタ及びコントロールレジスタを介して、ホストコンピュータにインタフェースする。これらを、慣用の方法でホストインタフェースバスコントローラによって構成し、ホストコンピュータのメモリ又は入力／出力スペース中に設ける。Rバスに接続される機能の例示：

１）一つのロケーションをRバスを介して書き込む際に、サンプリングクロックを周期化させ、ホストコンピュータのコマンドに従って、他のRバスロケーションからサンプリングされたデータの値を読出す、１グループとなっている８個のサンプラ。

２）ホストが特定のRバスロケーションに書込みを行う際に、データ値を変化させる、１グループとなっている８個のランダムアクセスメモリ。

３）各メモリロケーションを唯一のRバスロケーションに作成している設計部メモリ。ホストデータアクセスを行い、Rバスのアドレススペースへの読出し又は書込みオペレーションによって、ホストがアドレス指定された設計部メモリロケーションを読出し又は書込みすることができる。

他のこのような機能を容易に案出することができる。図４２に、Rバスの動作を示す。ロケーションを読出すために、リアライザシステムを作動させるホストコンピュータでランするプログラムは、アドレスを、ホストインタフェースバスアドレスレジスタにロードするとともに、“読出し”コマンドビットをホストインタフェースバスコントロールレジスタにセットする。その後、ホストインタフェースバスコントローラは、Rバス読出しサイクルを作動させる。１回に８ビットずつ、各々Rバスクロックの周期で、アドレスはRバスデータラインに与えられる。第一サイクルの間、バスコントローラは“同期”Rバスコントローララインに、Rバスサイクルが開始していることを表明する。その後、“読出し”Rバスコントロールライン及びRバスクロックが５回目の周期を成し、アドレス指定されたバスインタフェースコントロールロジックブロックが、その読出しオペレーションを完成させることができる。Rバスクロックが６回目の周期を成す間、アドレス指定されたバスインタフェースコントロールロジックブロックが、読出しデータを８個のRバスデータラインに伝送する。バスコントローラはこのデータを捕捉し、ホストインタフェースバスデータレジスタにロードするとともに、“コンプリート”コマンドビットを、ホストインタフェースバスコントロールレジスタにセットする。“コンプリート”ビットを認識するホストプログラムをセットし、データを読出し、“コンプリート”ビットをクリアする。ホストプログラムが“書込み”コマンドビットをセットし、書込まれるべきデータをホストインタフェースデータレジスタにロードすることを除き、ロケーションの書込みも同様である。バスコントローラは、５回目のクロック周期において“読出し”Rバスコントロールラインを主張することはなく、６回目の周期においてデータをRバスデータラインに伝送する。この時、データがアドレス指定されたバスインタフェースコントロールロジックブロックによって捕捉される。ロジックチップ中に構成されるバスインタフェースコントロールロジックブロックは、上述した動作に従って、有限状態マシンと、完全に慣用となっている方法で制御された機能を用いてRバスを接続するデータパスとを具えている。

【００５６】１.５.３　構成バス

構成バス及びその使用とオペレーションとを構成セクションで説明する。バスは、ホストインタフェースを介してホストコンピュータによって制御される。バスを、データレジスタ及びコントロールレジスタを介してホストコンピュータにインタフェースさせる。これらのレジスタを慣用の方法でホストインタフェースハードウエアによって構成し、ホストコンピュータのメモリ又は入力／出力スペースに設ける。ホストコンピュータにおいてランする構成プログラムにって、構成バスデータレジスタにロードされるデータを構成バスデータパスに伝送する。ホストコンピュータが構成バスコントロールレジスタに書込みを行う場合、ホストインタフェースハードウエアは構成バスクロックを１周期させる。

【００５７】１.５.４　コントローラ及びクロック発生

器のリセット
リアライザシステムのリセットコントローラは、2つのリセット信号を発生させる。システムリセット信号を、すべてのロジック及び相互接続チップのリセット入力ピンに接続する。ホストが主張する場合には、すべてのチップをリセットモードにし、構成の準備状態にする。慣用的な設計による1以上のプログラム可能なクロック信号発生器は、すべてのLチップのI／Oピンに分配される出力信号を有している。ホストは、その出力周波数を制御し、サイクルを停止させること、再びサイクルさせること、特定回数サイクルさせること、連続的にサイクルさせること等が可能である。ホストは、リアライザシステムにおいて実現される設計部のクロック発生器として使用される。クロック信号を制御することによって、設計部のオペレーションを制御する。設計部のリセット信号を、すべてのLチップのI／Oピンに接続する。設計部のリセット信号を、リアライザシステムにおいて実現される設計部をリセットする手段として用いる。これらの信号は、リアライザシステムによって実現される設計部との接続に利用することができる。特別な特性を入力設計ファイル中の回路網に組み込むことによって、入力設計部中の回路を、システムリセット又はクロックとして選定する。設計部リーダはこの特性を認識し、回路網を設計データ構造のリセット又はクロック回路網として特徴づける。設計変換システムの相互接続及びネットリスティング部分によって、この回路網をハードウエアの設計リセット信号又はクロック信号に接続されたI／Oピンに割当てる。

【0058】2　リアライザ設計変換システム
リアライザ設計変換システムは、設計部リーダ、基本要素コンバータ、パーティショナ、ネットリスティング及び相互接続システム、ERCGAネットリスト変換ツール及び構成ファイルコレクタを具えている（図43）。ここでは、入力設計ファイルを入力として用い、出力として構成ファイル及び対応テーブルファイルを作成する。これらは、リアライザハードウエアを構成、使用するための種々の応用に用いられる。入力設計を変換するために：
1）設計部リーダを用いて、設計部をメモリデータ構造に読込む。
2）設計部データ構造中の基本要素を、ホストEDAシステム固有の基本要素から、ERCGAネットリスト変換ツールと適合するネットリストファイル中に送信することのできるロジックチップ基本要素へ変換する。
3）パーティショナを用いて、各構成要素をどのロジックチップに対して使用するかを決定する。
4）ネットリスティング及び相互接続システムを用いて、リアライザハードウェアシステム中の各ロジックチップ及び相互接続チップに対するネットリストファイルを出力する。
5）ERCGAネットリスト変換ツールを繰り返して使用することによって、各ネットリストファイルを、対応する構成ファイルへ変換する。
6）簡単な方法である構成ファイルコレクタを用いて、各ロジック及び相互接続チップの構成ファイルから、この設計部の単一構成ファイルへ送信される構成データを捕捉し、これを用いてリアライザハードウエアを構成する。
ここで説明した設計変換のための方法を、注意したことを除いて、入力設計部のロジックゲートとフリップフロップとの組合せの変換に用いる。これらの方法の変形例を用いて、特定目的の基本要素を変換する。これらの変形例については該当するセクションにおいて説明する。

【0059】2.1　設計部リーダ
設計部リーダは、入力設計部ファイルを読出すとともに対応する設計部データ構造を構成する。

【0060】2.1.1　入力設計ファイルの要件
ホストEDAシステムによって作成される入力設計ファイルは、基本要素及びこれらの入出力ピンに関する記述と、2以上のピンを互いに相互接続するとともに、設計部の入出力端子と相互接続する回路網に関する記述とを有している。入力設計ファイルは、ネームのような、基本要素、ピン及び回路網に関連する情報も具えている。入力設計ファイルは、リアライザ設計変換システムが読出すことのできるように、基本要素の形態となっていなければならない。“基本要素”とは、ゲート、フリップフロップ又はメモリデバイスのような基本的ロジック素子である。設計者が特定することのできる基本要素によって規定される、より高レベルの構成を、リアライザシステムの読出し前に、EDAシステムによって構成基本要素に変える必要がある。入力設計部において許容される一組の基本要素の一例としては、以下のMentor Graplrics Quick Sim 基本要素がある。これは、好適例において読出される：
・25個までの入力端子を有する簡単なゲート（BUF,INV,AND,OR,NAND,NOR,XOR,XNOR）
・特別ゲート（DEL：ディレイ要素；RES：抵抗器；NULL：開放回路）・トライステート出力である無方向性伝送ゲート（XFER）
・記憶デバイス（LATCH 、レベル感応フリップフロップ又はREG、クロックされたフリップフロップ）
・メモリデバイス（RAM又はROM）

【0061】2.1.2　設計部データ構造
設計部リーダは、設計部データ構造を構成し、これを用いて基本要素を、ロジックチップネットリスティングに適した形態に変換し、基本要素をロジックチップサイズの区分に分割し、いかにしてロジックチップを相互接続するかを決定する。また、最終的には、設計部データ構造を、各リアライザロジックチップのネットリストファ

イルへと読出す。データ構造は、設計部の各基本要素、各ピン及び各回路網のレコードから成っている。各レコードは、関連に応じて、他のレコードに対するエンティティ及びリンク（すなわちポインタ）についてのデータを具えている。

・“基本要素”とは、ゲート、フリップフロップ又は、メモリデバイスのような基本ロジック要素である。

・各基本要素は、基本要素レコードによって表現される。基本要素レコードは、そのタイプ及び対象識別子のような基本要素に関するデータを有するとともに、他の基本要素とのリンクを有している。

・基本要素レコードは二重にリンクされたリストである。

・ピンとは、基本要素の入力接続又は出力接続である。

・基本要素のピンは、基本要素レコードと隣接して配置されるとともに、ピンネーム、ピンが反転されているかどうか、ピンの出力ドライブ等のピンに関するデータを有する、一連のピンレコードによって表現される。

・各基本要素は、一つの出力ピンのみを有しており、これを、任意のピンレコードとすることができる。

・“回路網”とは、相互接続されたピンの集合である。

・各回路網は、回路網レコードによって表現される。この回路網レコードは、その対象識別子のような回路網についてのデータを有するとともに、他の回路網へのリンクを具えている。

・回路網レコードを、二重にリンクされたリスト中に設ける。

・回路網のピンを、単一リンクの循環リスト中に設ける。

・各ピンレコードも、ピンの回路網へのリンクを有している。

・各回路網レコードは、回路網中の一つのピンへのリンクを有している。

図４４ａは、回路網の簡単な例を示しており、図４４ｂは、設計部データ構造を用いて回路網をどのように表現するかを示している。

【００６２】２.１.３　設計部リーダの方法論

設計部リーダは、入力設計ファイルから実現されるべき設計部を読出すとともに、対応設計部データ構造を構成することを目的としている。ここでの説明は、Mentor Graphics 設計ファイルに適合している。他も同様である。設計ファイルは、設計部中の各基本要素に対して、インスタンス（instance）と称するエントリを有している。設計ファイル中のインスタンスに取付けられた基本要素の特定のアスペクトについての情報が特徴である。以下に示す各工程の括弧内のネームは、好適例において用いられる実際のルーチンのネームである。

１）基本要素のレコードと、設計ファイル中の各基本要素に対するメモリ内データ構造中のピンのレコードとを以下のように作成する：各設計ファイルの基本要素の各インスタンスは：基本要素のタイプが何であるかを読出す（ｇｅｔ＿ｄｆｉ＿ｍｏｄｅｌ＿ｔｙｐｅ）。ユーザが規定したこの基本要素の配置についての情報が存在する場合には、これを“１チップ”特性から得る。；設計ファイルインタフェースを用いて、より高度な非基本的インスタンスをサーチする。このインスタンスはこの基本要素を具え、同様に特性を調べる（ｇｅｔ＿ｄｆｉ＿１ｃｈｉｐ）。インスタンスの各ピンは：ピンのネーム等の、ピンに関する任意の特性を捕捉する（ｇｅｔ＿ｄｆｉｐｉｎ＿ｉｎｆｏ）。次のピンへ。メモリ内設計データ構造中のレコードを、この基本要素及びピンに割当てる（ａｌｌｏｃ＿ｐｒｉｍ＿ａｎｄ＿ｐｉｎｓ）とともに、基本要素レコードを満たす。各々のピンは：ピンレコードを満たす。（設計ファイル中の接続された回路網の対象識別子ナンバを記憶し、トラックの識別子のナンバを、最大に維持する。）次のピンへ。次の設計ファイルインスタンスへ。ポインタのテーブル（ｎｅｔ＿ｔａｂｌｅ）を、ピンレコード(ピンポインタ)に割当てる。各構成し得る回路網に対するピンレコードに対象識別子ナンバで索引を付ける。最初はＮＵＬＬとする。上記最大識別子ナンバに従ってテーブルを作る。

２）各回路網のピンレコードをリンクし、以下のような、循環的にリンクされたリストとする：メモリ内データ構造中の各基本要素レコードにおいて：各ピンレコードは：‘ｉｄ’を、このピンの接続された回路網の対象識別子ナンバとすると、ｎｅｔ＿ｔａｂｌｅ〔ｉｄ〕が非ＮＵＬＬピンポインタを有している場合、これをこのピンレコードの“ｎｅｘｔ　ｐｉｎ”リンクへコピーする。このピンに対するピンポインタを、ｎｅｔ＿ｔａｂｌｅ〔ｉｄ〕に入れる。次のピンへ。次の基本要素へ。

３）各回路網に対する回路網レコードを以下のように作成する：ｎｅｔ＿ｔａｂｌｅ中の各ピンポインタは：回路網レコードを割当てる。リンクを用いて、回路網レコードを、ピンポインタの指示するピンに接続する。対象識別子ナンバを用いてアドレス指定を行うことにより、設計ファイルインタフェースから回路網に関する情報を得る（ｄｆｉ＿ｇｅｔ＿ｎｅｔ,ｇｅｔ＿ｄｆｉ＿ｎｅｔ＿ｉｎｆｏ）。この回路網におけるピンレコードの循環リスト中の各ピンについて：回路網レコードに指示する。次のピンへ。循環リストを閉じる：最終ピンを最初のピンにリンクさせる。次のピンポインタへ。ｎｅｔ＿ｔａｂｌｅ記憶機能を解除する。

４）内部メモリ設計データ構造が終了し、設計部変換プロセスの後段が必要とする実現されるべき設計部についてのすべてのデータを表示する。

【００６３】２.２　基本要素コンバータ

基本要素変換は、Mentor Graphics Quick Sim 基本要素のような、ホストが特定する基本要素からの設計部データ構造中の基本要素を、ＥＲＣＧＡネットリスト変換ツールと適合させ、ネットリストファイル中に送出される

ロジックチップ指定の基本要素に変換することを目的としている。この変換のいくつかは、簡易且つ直接的なものであり、単に、基本要素のタイプ及びピンネームのみを置換えるのみである。その他の変換はかなり複雑である。以下に示す特定の引用例は好適例のためのものであり、Mentor Graphics 入力設計ファイル中に存在するMentor Graplics Quick Sim のホストが特定する基本要素と、Xilinx ＬＣＡロジックチップが特定する基本要素とを使用する。設計部のゲートが、ロジックチップの特定するゲート基本要素中で許容されるよりも多くの入力端子を有している場合、このゲートを等価な機能を有するゲートの回路網で置換える。このゲート回路網の各々は許容数の入力端子を有している。このような置換を行うために、ゲートの基本要素レコード及びピンレコードを取り除き、新しいゲートの基本要素レコード及びピンレコードと、回路網中の新しい回路の回路網レコードとを加え、置換えられたゲートに接続されているピン及び回路網のピンレコード及び回路網レコードにリンクする（図４５ａ）。設計部のフリップフロップが、ロジックチップの特定するフリップフロップ基本要素において利用することのできない機能を有している場合、フリップフロップを等価な機能を有するゲートの回路網で置換える。まず第１に、回路網を解析し、機能を常に一定の値ではない回路網に接続するかどうかを調べる。例えば、ホストが特定する基本要素ＲＥＧを、常に一定の値ではない活動回路網に接続されたダイレクトクリア入力及びダイレクトセット入力の両方を用いて使用する場合に、メモリ内設計部データ構造における基本要素を、必要に応じて機能する７４７ＴＴＬフリップフロップロジックパートに用いられるのと類似のゲートの回路網で置換える。しかし、ダイレクトセット入力を、グランド回路網のような常にロジック値ゼロの回路網に接続する場合、すなわち、例えばグランド回路網に接続された１つの入力端子を有するＡＮＤゲートの場合には、ダイレクトクリアのみが実際に必要とされ、その代わりにロジックチップＤフリップフロップを代用する。Ｓ＿ＲＡＭ基本要素は、アドレス入力端子、双方向性データポート、読出しイネーブル及び書込みイネーブルを有するランダムアクセスメモリである。ＲＡＭ基本要素を、１以上のリアライザ設計部メモリモジュール中に作成する。基本要素変換ソフトウエアは、Ｓ＿ＲＡＭを利用可能な設計部メモリ構成と直接的にマッチする１以上のＸ＿ＲＡＭ基本要素に変換する。Ｓ＿ＲＯＭ（出し専用メモリ）基本要素は、イネーブル入力端子が存在せず、且つ、ＲＯＭの内容を含むファイルを付加していることを除き、Ｓ＿ＲＡＭと同様のものである。Ｓ＿ＲＯＭ基本要素を設計部メモリ構成と直接マッチする１以上のＸ＿ＲＯＭ基本要素に変換する。Ｘ＿ＲＯＭは読出しイネーブル入力端子を有しているが、書込みイネーブルを有していない。内容ファイルのパスネーム及びもとのＳ＿ＲＯＭに対するそのロケーションを、各Ｘ＿ＲＯＭ基本要素を用いて記憶する。リアライザハードウエアをこの設計部を用いて構成する場合、構成システムがパスネームを用いて、Ｘ＿ＲＯＭ内容を取り出すとともに、これらをホストインタフェースを介して設計部メモリにロードする。分離入出力データポートを有するＳ＿ＲＡＭは同様に操作される。しかし、これはMentor Graphics Quick Sim 基本要素中には設けない。オリジナル設計部のピン及び回路網は、初期特性すなわち“ｉｎｉｔｓ”を送信し、ある場合に持続的に幾つかの初期値を送信していることを示している。既知の値（０又は１）である持続的な初期特性がリアライザシステムによって観測され、ピン又は回路網を適切な“グランド”（すなわち、ロジック値０）又は“ＶＣＣ”（すなわち、ロジック値１）の回路網に接続する。特定のMentor Graphics の場合では：

・Ｔ、Ｘ、Ｒ及びＺの初期特性を無視する。ＯＳＦ（＝０＝ＯＳ）又は１ＳＦ（＝１＝1Ｓ）のみを観測する。

・回路網、すなわち、回路網の任意の出力ピンのＯＳＦ又は1ＳＦによって、出力ピンをグランド又はＶＣＣ回路網の一部分とする。

・入力ピンのＯＳＦ又は１ＳＦとによって、このピンを絶縁し、グランド又はＶＣＣ回路に接続する。

【００６４】オリジナル設計部の出力ピンは種々の強さのドライブを伝達し、シミュレータによって形成されるべき出力構造のタイプを示す。リアライザシステムは、基本要素変換の際に、幾分これらの強さを観測する。出力端子を、ハイのときにドライブの強さが零であり、ローのときにドライブの強さが強であるように特徴づける場合、出力端子はオープンコレクタとして識別され、これを他の同様の出力端子及びレジスタに、ロジック設計者が“ワイヤード・アンド”回路網（図４５ｂ）と称する形態で接続するのが正当的である。同様に、ローのときにドライブの強さが零であり、ハイのときにドライブの強さが強である出力端子はオープンエミッタであり、“ワイヤードオア”を形成するのに用いられる。最終的にイネーブルされなければ、ＸＦＥＲ基本要素の出力ピンはドライブを有さず、他のＸＦＥＲ出力端子及びレジスタと配線され、“トライステート”回路網を形成する（図４５ｃ）。これらの構造のすべては、基本要素変換システムによって認識され、トライステート回路網のセクションにて説明したように、等価な機能を有する積の和のロジック回路網に変換される。特定のMentor Graphics の場合：

・Ｘ－ステートドライブの強さを無視する。

・１以上のＸＦＥＲ出力端子を回路網に接続することができるが、他の出力端子を接続することはできない。例外としては、入力ピンをグランド又はＶＣＣ回路網に接続しているＲＥＳ（抵抗器）を接続することもできる。ＸＦＥＲがイネーブルされない場合、回路網値はロジック値ゼロであり、ＶＣＣに接続されたＲＥＳを接続しな

い場合には、ロジック値1となる。1より多くのXFERをイネーブルする場合、結果は論理的ORとなる。
・OC/OE出力端子(SZ/ZS)は、同様のドライバを用いても駆動することのできる回路網のみを駆動することができる。駆動されていない場合、OC回路網はハイとなり、RESを接続しているかどうかを無視してOE回路網はローとなる。
・RZ、ZR、RSS、SR又はZZ出力ドライブを有する基本要素を、エラーなしで除去する。
・以下の出力回路網の条件によって致命的な誤りが生じる:すなわち、1より多くのストロング(strong)、ストロング及び抵抗器、1より多くの抵抗器、XFER及びストロング、XFER及びSZ,XFER及びZS、抵抗器を有していないSZ又はZS、ストロングを有するSZ又はZS、SZ&ZSである。
【0065】Mentor Graphics ホスト及びXilinx LCAを有する好適例の基本要素を変換するための特別な手続は、以下に示すとおりである(サブルーチンのネームが各ヘッダの後に続いている):
1)ホストが特定する基本要素のLCA基本要素への初期変換(convert_s_to_x)。ホストが特定する基本要素は、上述のMentor GraphicsQuick Simセットから成り、'S_'プレフィクスを用いてネームが付けられる。LCAが特定する基本要素は、Xilinx.xnf仕様から成り、'X_'プレフィクスを用いてネームが付けられる。各基本要素は:S_INVの場合、X_INVと置換え、ピンのネームを置換える。S_BUSの場合、X_BUSと置換え、ピンのネームを置換える。S_RESの場合、X_BUF、RRドライブと置換え、ピンのネームを置換える。S_DELの場合、IN&OUT回路網を結合する。S_AND、S_NAND、S_OR、S_NOR、S_XOR、S_XNORの場合、X_AND、X_NAND、X_OR、X_NOR、XXOR、X_XORと置換え、ピンネームを置換える。(25ピンよりも多い場合、エラー)S_REGの場合、X_DEFと置換え、ピンネームを置換える。S_LATCHの場合、X_DLATで置換え、ピンネームを置換える。S_XFERの場合、後にまでピンネームを後にまで残しておく。S_NULLの場合、ピンネームをデリートする。S_RAM又はS_ROMの場合、ピンネームを後にまで残しておく。次の基本要素へ。
2)初期特性の処理(get_init)。メモリ内設計部データ構造中の2つの回路網は特別なものである。:すなわち"gnd"(ロジック値0)及び"VCC"(ロジック値1)である。各回路網は:回路網の初期特性がOSFである場合、gnd回路網が、初期特性がOSFであることを認識できない場合、次の回路網へ、認識できる場合には、この回路網をgnd回路網と組合せ、次の回路網へ。回路網の初期特性が1SFである場合、VCC回路網が、初期特性が1SFであることを認識できない場合、次の回路網へ、認識できる場合には、この回路網をVCC回路網と組合せ、次の回路網へ。各出力ピンは:ピンの初期特性がOSFの場合には、gnd回路網が、初期特性がOSFであることを認識できない場合には次の回路網へ、認識できる場合には、この回路網をgnd回路網と組合せ、次の回路網へ。ピンの初期特性が1SFの場合には、VCC回路網が、初期特性が1SFであることを認識できない場合には、次の回路網へ、認識できる場合には、この回路網をVCC回路網と組合せ、次の回路網へ。次のピンへ。次回路網は:ピンレコードを、リスト中に入れる。各入力ピンは:ピンの初期特性が0SFであり、且つこの回路網がgnd回路網でない場合、ピンを回路網から切り離し、gnd回路網に接続する。ピンの初期特性が1SFであり、且つこの回路網がVCC回路網でない場合、ピンを回路網から切り離し、VCC回路網に接続する。次のピンへ。次の回路網へ。
3)すべての出力ピンをチェックし、リアライザシステムのドライブの強さに影響を与えないで基本要素を取り除くとともに、常にイネーブル又はディゼーブルされているXFERを取り除く。各基本要素は:出力ピンがドライブSS、RR、SZ又はZSを有していない場合、次のピンへ。出力ピンがRZ、ZR、RS、SR又はZZを有している場合には、出力ピンを切り離し、除去する。出力ピンがS_XFERである場合:E0(イネーブル)ピンが常にローである場合には、基本要素をデリートする。E0ピンが常にハイである場合、BUFを代用する。次の基本要素へ。
4)不法なマルチ出力接続を選別し、ワイヤードOR,ワイヤードAND及びトライステート回路網及びこれらのドライバを識別し、変換する(wired- nets)。各回路網は:ピンレコードを、リスト中に入れる。XFER出力ピン、入力ピン及びnon-XFER出力ピンを数える。これらのピンは、ストロング(strong)かつレジスティブ(resistive)なSZ(オープンコレクタ)又はZS(オープンエミッタ)である。ストロングを有する又はドライブ強度を有しない唯一の出力ピンの場合、次の回路網へ。1以上のレジスタを接続する場合、すべてのレジスタを、'VCC'(プルアップ)又は'ground'(プルダウン)のいづれか一方に接続していることを確認し、いづれかを記憶する。以下の場合には、エラーであり、イグジット(exit)する:1より大きなストロング、1より大きなレジスタ。XFER及びストロング、XFER及びSZ、XFER及びZS。レジスタを有していないSZ又はZS、ストロングを有するSZ又はZS、SZ及びZS。1ストロング且つ1レジスタの場合には、レジスティブドライブを有する基本要素をデリートする。1より大きなSZの場合:(オープンコレクタ ワイヤードAND)各出力ピンは:レジスタ

の場合、出力ピンがプルアップであることを確認し、これをデリートする。レジスタでない場合には、このピンののドライブをストロングとし、X－ＩＮＶを構成し、この入力端子を出力ピンに接続するとともに、この出力端子を回路網に接続する。次のピンへ。回路網を“フローティングハイ”トライステート回路網として特徴付け、ＯＲ／ＮＯＲゲートを用いて、相互接続によってこれを構成する。１より多くのＺＳの場合：（オープンエミッタ　ワイヤードOR)各出力ピンは：レジスタの場合：ピンがプルダウンであることを確認し、その後、これをデリートする。レジスタでない場合は、ピンのドライブをストロングにする。次のピンへ。回路網を“フローティングロー”トライステート回路網として特徴付け、インタコネクタによって、ORゲートを用いこの回路網を構成する。０より多くのＸＦＥＲ且つレジスタ無し、又は、プルダウンの場合：（トライステート“フローティングロー”）各Ｓ－ＸＦＥＲは：ＡＮＤ　１０を構成するＸＦＥＲ　Ｅ０（又はＥＮＡ）及びＡＮＤ１１を構成するＸＦＥＲ１０を用いて、Ｓ－ＸＦＥＲをＸ－ＡＮＤに変換する。次のＳ－ＸＦＥＲへ。任意のレジスタ基本要素をデリートする。回路網を“フローティングロー”トライステート回路網として特徴付け、ＯＲゲートを用いて、相互接続によってこれを構成する。０より多くのＸＦＥＲ基本要素且つ、プルアップの場合：（トライステート“フローティングハイ”）１個のＳ－ＸＦＥＲ基本要素の場合：ＮＡＮＤ１０を構成するＸＦＥＲ　Ｅ０（又はＥＮＡ）及びＮＡＮＤ１１を構成するＸＦＥＲ１０を用いて、Ｓ－ＸＦＥＲをＸ－ＮＡＮＤに変換し、反転する。１より多くのＳ－ＸＦＥＲ基本要素の場合：各Ｓ－ＸＦＥＲは：ＡＮＤ１０を構成するＸＦＥＲ　Ｅ０（又はＥＮＡ）及びＡＮＤ１１を構成するＸＦＥＲ１０を用いて、Ｓ－ＸＦＥＲをＸ－ＡＮＤに変換し、反転する。次のＳ－ＸＥＲへ、レジスタ基本要素をデリートする。回路網を“フローティングハイ”トライステート回路網として特徴付け、ＯＲ／ＮＯＲゲートを用いて、相互接続によってこれを構成する。次の回路網へ。

５）等価な機能を有するゲート回路網を用いて、ＬＣＡが特定するゲート基本要素中に許容されるよりも多くの入力ピンを有する任意のゲートを置換する。ゲート回路網の各々は、許容数の入力端子を有している。(wide-gates)各基本要素は：そのゲート及び入力端子が５よりも多く、２５以下である場合（XC3000ロジックチップを用いるものと仮定する）：同一種類の最終出力ゲートを構成する。その出力端子を、オリジナル出力端子及びコピー特性に接続する。必要とされる、より小さな入力ゲートの各々は：（ＡＮＤまたはＮＡＮＤオリジナル等のためのＡＮＤを用いて）ゲートを割当てる。ゲートの出力端子を最終ゲート入力端子に接続する。ゲートの入力端子を本来の入力端子に接続する。次のゲートへ。オリジナルのワイドゲートをデリートする。次の基本要素へ。

６）フリップフロップの機能をチェックするとともに、ＬＣＡの制約に合致するように配置する。XC3000シリーズを用いる場合、フリップフロップはダイレクトクリアを有するがダイレクトセットは有しておらず、従って両者を有するわけではない。すべてのＳ－ＤＥＦは、セット及びクリアのためのピンを有しているため、基本要素は、ピンを有しているといないとにかかわらず、置換える必要がある。XC3000はラッチをサポートしないため、ラッチを、等価なゲート回路網で置換える必要がある（ｆｌｏｐｓ＿ｆｏｒ＿３Ｋ）。各基本要素は：基本要素がＤＬＡＴ又はＤＥＦである場合：各ピンを記憶するとともに切離す。ＳＤ及びＲＤ回路網がゲートを介して、直接又は非直接的に‘ｇｒｏｕｎｄ’又は‘ＶＣＣ’であるかをチェックすることによって、ＳＤ及びＲＤが常にローであるかどうかを見出す。各基本要素がＤＬＡＴである場合：ゲートの回路網中に組込み、必要な場合にのみ、ＳＤ及び／又はＲＤのためのゲートを具えるラッチを構成する。オリジナル基本要素及びピンレコードをデリートする。各基本要素がＤＬＡＴでなく、ＤＥＦである場合：ＳＤが常にローの場合、ＳＤを用いずにＸ－ＤＥＦを構成し、これを接続する。ＳＤがローでなく、ＲＤがローの場合、入力端子及び出力端子にＸ－ＩＮＶを用いてＸ－ＤＥＦを構成し、これを接続し、Ｘ－ＤＥＦのＲＤピンを、ＳＤ回路網に接続する。ＳＤが常にローではない場合、６個の３－入力ＮＡＮＤ及び２ＩＮＶの回路網に組込み、TTL7474 と同様に、セット及びクリアを有するＤＥＦを構成する。オリジナルの基本要素をデリートする。次の基本要素へ。

７）Ｓ－ＲＡＭ及びＳ－ＲＯＭを、Ｘ－ＲＡＭ及びＸ－ＲＯＭに変換する。各基本要素は：基本要素がＳ－ＲＡＭ又はＳ－ＲＯＭである場合：そのハイト（height）（ワード数）を、アドレスピン（ハイト＝２～ピンカウントの累乗）及び、データピンナンバと等しいアドレピンの幅をカウントすることによって決定する。各利用可能な設計部メモリ構成は：Ｓ－ＲＡＭ／ＲＯＭハイトを、設計部メモリハイトで割算し、必要なモジュールの行数を得る。Ｓ－ＲＡＭ／ＲＯＭの幅を設計部メモリの幅で割算することにより、必要なモジュールの列数を得る。この構成のために必要なモジュールの総数は行掛ける列である。次の構成へ。必要なモジュール数が最小となる構成を選択する。行よりも多くのモジュールを必要とする場合、モジュールの各行に対する出力端子とハイオーダのアドレス回路網に接続された入力端子とを用いて、デコーダの基本要素及び回路網を構成する。各行は：（Ｘ－ＲＡＭのみ）２個の入力端子を用いて書込みイネーブルのためのＡＮＤゲートを構成する：２個の入力端子とは、この行のデコーダ出力端子及びＳ－ＲＡＭ書込みイネーブルである。２つの入力端子を用いて、行読出しイネーブルのためのＡＮＤゲートを構成する：こ

の2個の入力端子とは、この行のデコーダ出力端子及びS-RAMリードイネーブルである。次の行へ。モジュールの各行について、：各コラムは：X-RAM/ROM基本要素を構成するとともにその構成を記憶する。X-ROMの場合、そのファイルネーム及び行数及び列数を記憶する。X-RAM/ROMの読出し及び書込みイネーブルピンを、この行の(X-RAMのみの) 読出し及び書込みイネーブルピンに接続（又は、この行が1つのみの場合には、S-RAMのイネーブルピンに接続）する。X-RAM/ROMのアドレスピンを、より低オーダのアドレス回路網に接続する。X-RAM/ROMのデータピンを、この列に対応する一組のデータピンに接続する。次の列へ。次の行へ。オリジナルのS-RAM/ROM基本要素をデリートする。次の基本要素へ。

【0066】2.3 パーティショナ

リアライザハードウェアはユニット及びサブユニットの階層から成っている。すなわち、ボードは論 理チップを具え、ボックスはボードを具え、ラックはボックスを具える等である。各ユニットは、ユニット固有のロジック及び他のユニットとの相互接続のための容量を有している。実現すべき設計部をこの階層に応じて区分化し（すなわち、サブ分割し）、基本要素の多重クラスタとする。各ボックスのロジック及び接続容量に応じて作成された一組のボックス区分を設ける。これらの区分の各々を、ボードのサブ区分に分割する等により、単一の論理チップにプログラムするのに十分な程小さな区分に分割する。同様の分割化方法を、階層の各レベルに順次適用する。分割化の目的とは：

1）各基本要素を、ボックス、ボード及びロジックチップに割当てること、

2）ユニット（ボックス、ボード又はロジックチップ）の相互接続能力の下、区分と接続している回路数を保持すること、

3）ユニットの限界内で分割化に用いられるロジックの量を保持すること、および

4）区分の総数すなわち使用されるユニットの総数を最小にすることである。

【0067】2.3.1 分割化方法

ここで説明する好適分割化方法は、“カット(CUT)回路網”（クラスタの外部における基本要素の接続）の数が最小であり、お互いに高密度で相互接続されたロジック基本要素を密集させるプロセスに基づくものである。各クラスタは、ボックス、ボード又はLチップに対応する区分である。前記プロセスは、以下において指摘する、相当な改良を伴うPalesko 及びAkers の従来の区分化方法(Chet A. Palesko, Lex A. Akers, “Logic Partitioning for Minimizing GateArrays”,IEEE Trans, CAD, NO.2, pp. 117 ～121, April 1983)に基づいている。すべての基本要素を初めから見えているクラスタに、割当てられていない基本要素から成る“ナル（null）クラスタ”を設ける。まず、ナルクラスタからシード（seed）クラスタを選択し、その後、これをリピートし、すべてのナルクラスタ基本要素の “利点”を計算し、最も大きな利点を有する基本要素を選択することによって各クラスタを形成する。基本要素の利点が大きくなるにつれて、ロジッククラスタに組込むのに適したものとなる。

【0068】2.3.2 有利な機能

部分的な利点とは、この基本要素をクラスタに組込む場合にこのクラスタのカット回路網の数がどのように変化するかに基づくことである。ユニットを最大相互接続可能性以下に保持するために、クラスタのカット回路網総数をカウントする必要がある。基本要素のピンを具えている各回路網は垂直となっており、基本要素を組込むものを仮定すると、 閉回路網、“多数カット”の回路網又は“単一カット”の回路網のいずれかに分類される。一つのみの接続をクラスタの内側に設けると単一カット回路網であり、1より多くの接続をクラスタの内側に設けると多数カット回路網となる。閉回路網は、全体がクラスタ中に含まれる回路網をいう。図46は、クラスタと、3個の回路網S，M及びEによって接続された5個の基本要素とを示すとともに、影を付けた基本要素をクラスタ中に移動させるとどうなるかを示している。クラスタのカット回路網数を1個増加させると、回路網Sは、単一カット回路網となり、カット回路網数を1個減少させると、回路網Eは、閉回路網となる。回路網Mは、クラスタのカット回路網数を増加、減少させても多数カット回路網であり、このため無視される。クラスタ回路網の変化は、単一カット回路網と閉回路網との差である。すなわち：

〔クラスタカットの変化〕＝〔単一カット回路網〕－〔閉回路網〕

好ましい有利な機能とは、各基本要素の個数を定め、クラスタに組込むには、どの基本要素を選択するのが最適かを決定することである。最大数のピンと最も強固に接続された基本要素を選択するのが最適である。この機能は、Palesko andAkers の分割化に関する最初の有利な機能である。すなわち：

〔クラスタカットの変化〕＞0の場合：〔利点〕＝〔基本要素のピン数〕／〔クラスタカットの変化〕

〔クラスタカットの変化〕≦0の場合：〔利点〕＝〔－（クラスタカットの変化）＊100〕＋100＋〔基本要素のピン数〕

この基本要素をクラスタに組込む場合、クラスタカットの数が増加する。多くのピンを具えれば具える程、加えるカット回路網の数が少なければ少ない程優れている。クラスタのカット数が減少する場合、減少の程度は100倍となり、100が加えられ、カットを減少させない、いかなる基本要素の利点よりも大きな利点が得られることを保証している。クラスタカットが減少している

場合、ピンの数を増やし、接続を断ち切ることにより、基本要素を、更に多くのピント結び付ける。好適な方法で用いられる改良とは、クラスタカットが増加する場合に、ピン数の項をピン数／カット変化の比に加えることである。この変更は、前記比が等しい場合に、より多くのピンを有する基本要素を選択することによって、初期シード選択を改善することができる。前記比を１０倍することにより、ピン数単独よりも効果的にする。このことは、好適且つ有利な機能である。すなわち:
〔クラスタカットの変化〕＞０の場合：〔利点〕＝（〔１０＊（基本要素のピン数）〕／〔クラスタカットの変化〕）＋〔基本要素のピン数〕
〔クラスタカットの変化〕≦０の場合：〔利点〕＝〔－（クラスタカットの変化）＊１０００〕＋１００＋〔基本要素のピン数〕
【００６９】２.３.３　クラスタの構成
初めに、すべての基本要素をナルクラスタ中に配置する。ユーザは、選択するＬチップ、ボード等を表示する特性を入力設計部に付加することによって、基本要素を特定のクラスタ中に予め配置することができる。この時、これらの予め配置された基本要素は、クラスタ情報に関数するシード配置としての役割を果たす。このことによって、ユーザは、タイミング感知基本要素又は他の高優先基本要素を集めることができ、また、高優先基本要素に強固に接続されている他の基本要素を集めることによって、分割化の結果を変えることができる。各々の新しいクラスタに関して、始めに、新しいクラスタの未配置の基本要素の利点を計算し、基本要素レコード中に記録する。予め配置を設けない場合、最も有利な基本要素（すなわち、最も大きな利点を有するもの）を、クラスタの初期シード基本要素として選択する。最も有利な基本要素の各々をクラスタ中に移動させた後、組込まれた基本要素と同一の回路網におけるピンを有する基本要素のみが、利点を再び計算する。他の基本要素は移動によって影響を受けないので、クラスタのこれらの利点に変わりはない。従って、クラスタが一杯になるまで、新しい最も有利な基本要素をクラスタ中に移動させる。クラスタが一杯となる時を決定することは、ロジック容量及び相互接続（すなわち、クラスタカット回路網）の両方に依存している。基本要素を、クラスタ中に移動させる場合、常に、基本要素によってクラスタ中のゲート数が増加する。しかし、基本要素によって、カット回路網は、必ずしも増加しない。減少することもあり得る。Palesko and Akers の方法による相互接続の限界に達する場合、基本要素がロジック容量又は相互接続の限界を超えないならば、最大よりも少ない利点を有する基本要素をクラスタ中に移動させることができるが、局所的相互接続の最大を超える場合には基本要素をクラスタ中に移動させることはできない。ここで説明した方法を、以下の点において改良する。すなわち：マーカ（marker）のアレイを設ける。各マーカに対して、マーカは、起動可能である。基本要素を１個ずつクラスタ中に移動させる。各々の移動の後、クラスタカット回路網の数をチェックする。クラスタカット回路網の数がユニットの最大利用可能相互接続機能以下の場合、移動を、相互接続が可能なものとして認識される。最大ロジック容量の限界に達した場合、最後の移動は、相互接続可能なものとは認識されず、最後の移動が相互接続可能となるまでは移動は行われない。ユニット（ラック、ボックス又はボード）をサブユニット（ボックス、ボード又はＬチップ）に分割化するために：予め配置されていないすべての基本要素をナルクラスタ中に移動させる。各クラスタは：各ナルクラスタ基本要素の利点を計算するとともに記憶する。起動カウンタの数をゼロにする。〔クラスタ基本要素カウント〕＜〔最大ロジック容量〕の場合、移動カウンタをインクリメントする。最も有利な基本要素をクラスタ中に移動させる。どの基本要素を移動させるかを、移動カウンタに記録する。〔クラスタカット回路網〕＜〔最大相互接続容量〕の場合、ｍｏｖｅ〔移動カウンタ〕＝ＯＫをマークする。〔クラスタカット回路網〕≧〔最大相互接続分量〕の場合、ｍｏｖｅ〔移動カウンタ〕＝ＮＯＴ　ＯＫをマークする。このクラスタに接続された回路網の利点を計算する。次のリピートへ。ｍｏｖｅ〔移動カウンタ〕＝ＮＯＴ　ＯＫの場合、ｍｏｖｅ〔移動カウンタ〕に記録された基本要素をクラスタから外へ移動させる。移動カウンタをデクリーメントする。次のリピートへ。次のクラスタへ。区分化のプロセスは、すべての基本要素を首尾よくクラスタ中に配置するまで又はすべてのクラスタが一杯となるまで続き、プロセスが終了する。好適例の全設計部を区分化するために：ラッチレベルのボックス、すなわち、各ボックス毎に１個のクラスタに分割化する。この際：
〔最大ロジック容量〕＝〔全ボックス及び最大相互接続容量〕＝〔ボックス毎のＹ－Ｚパス〕
を用いている。各ボックスクラスタは：ボックスレベルのボードへの区分化、すなわち、各ボード毎に１個のクラスタに分割化する。この際：
〔最大ロジック容量〕＝〔全ボード及び最大相互接続容量〕＝〔ボード毎のＸ－Ｙパス〕
を用いている。次のボックスクラスタへ。各ボードクラスタは：ボードレベルのＬチップへの区分化、すなわち、各Ｌチップ毎に１個のクラスタに分割化する。この際：
〔最大ロジック容量〕＝〔Ｌチップ及び最大相互接続容量〕＝〔Ｌチップ毎のＬ－Ｘパス〕
を用いている。次のボードクラスタへ。
【００７０】２.３.４　容量の限界
この方法に用いられる最大ロジック容量の限界の決定は、使用するロジックチップの特性に依存している。Xilinx製のＬＣＡを論理チップとして使用する場合、これ

らは、構成可能な論理ブロック(ＣＬＢ)を基礎にしている。ＣＬＢの各々によって、多くのゲート及びフリップフロップを具現化することができる。ＣＬＢの数は、ゲート及びフリップフロップ機能、どの位多くの機能を設けるか、どのくらい多くのピンを有するか、どのように相互接続するかに依存している。分割化前に設計部をＣＬＢの形態に変換する場合、ＣＬＢを分割化された基本要素とし、ロジック容量の限界はＬＣＡ中のＣＬＢの数に基づいている。分割化前に設計部をＣＬＢの形態に変換しない場合、ゲートを、分割化された基本要素としてロジック容量の限界は、ＬＣＡに適合すべきゲートの数に基づいている。ゲートが消費する容量の程度に応じて、ゲートの重みをかけ、分割化の結果を改善する。各クラスタを構成するのに用いられる限界は、必ずしもすべて同一である必要はない。ユニット間でロジック及び相互接続容量特性を変える場合、適切に限界を設定してこれらユニットのクラスタを構成する。

【００７１】２.３.５　リアライザの区分化

プロセスを分割化することによって、設計部の各基本要素に対する３ナンバボックス／ボード／チップロケーションが得られる。このロケーションは、設計部データ構造の基本要素レコードに記憶される。このことによって、設計部回路網の各基本要素は、Ｌチップ、ボード、及びボックスに亘ってトレースすることができる。ネットのタイミングは、システムにおける回路網をトレースするとともに、相互接続クロスバーチップ及び論理チップを介して、ディレイを加算することによって評価される。相互接続の段階では、回路網中に具えられた種々のボックス／ボード／チップ基本要素の組合せ総数に基づき、ネットリストをオーダする。このようにして、相互接続は最も複雑な回路網から最も複雑でない回路網までを保証する。最終的に、回路網及び回路網レコードの基本要素が、Ｌチップ及びクロスバーチップに亘って回路網を明確に作成する情報を具えているので、局所的な図式的ロジック変化を再度区分化する必要はなく、変化させられた回路網を具えるチップを更新する必要があるだけである。これによって、設計部を再度区分化せずに設計部を変換させることができる。

【００７２】２.４　ネットリスティング及び相互接続システム

リアライザネットリスティング及び相互接続変換システムは、入力設計に応じて、リアライザハードウェアを構成するのに用いる、リアライザシステム中の各ロジックチップ及びクロスバーチップに関するネットリストファイルを構成することを目的としている。どのようにして部分的クロスバー相互接続をネットリストすべきかの決定を、次の３段プロセスの総合によって行う。

ステージ１：ステートメントは、各基本要素毎に設計データ構造中のすべてのロジック基本要素に関するロジックチップネットリストファイルに送信される。

ステージ２：完全に単一ロジックチップ中に含まれているトライステート回路網の加算ゲートに関するステートメントは、各回路網毎に送信される。

ステージ３：より多くのロジックチップ間を通る回路網の相互接続をネットリストする。各カット回路網毎に、すべてのチップにおけるこの回路網のすべての相互接続バッファ及び、クロスバーチップにおけるこの回路網の加算ゲートに関するステートメントを送出する。このプロセスの一部として、いかにして回路網を相互接続するかを明確に決定する。このプロセス自体は４個のステージを有している。

ステージ３ａ：どのようにして回路網が各クロスバー間を通り、且つ、どこにロジックチップドライバ及びレシーバを配置するかを示しているトリー(tree)を構成する。

ステージ３ｂ：クロスバーチップの各組の回路網を相互接続する能力を評価する。

ステージ３ｃ：この回路網を相互接続するクロスバーチップの最適な組を選択する。

ステージ３ｄ：組の選択及びトリー構造に基づき、バッファ及び加算ゲートに関するステートメントをロジック及びクロスバーチップネットリストファイルに送出することによって、相互接続をネットリストする。

このセクションでは、各ステージに用いられる技術について説明しており、完全な相互接続及びネットリスティング手続について詳細な規定と、二つの詳細な回路構成例を記述している。

【００７３】２.４.１　シンプルな回路網及びトライステート回路網の相互接続構造

シンプルな回路網は、単一のドライバのみを有する回路網である。ドライバを有するソースＬチップは、信号を、階層の上方へ向かってすべてのレシーバに及んでいるクロスバーチップに伝達する。レシーバを駆動するためのパスを階層の下方に向かって接続し、すべての受信Ｌチップを駆動する。図４７は、シンプルな回路網の相互接続を示しており、詳細については以下に説明する。

トライステート回路網は、２以上のトライステート、オープンコレクタ又はオープンエミッタドライバによって駆動される回路網である。このことは、設計部データ構造中においては、２以上のドライバ(出力ピン)を有する単一の回路網として示される。各ドライバを、基本要素変換の間にドライバを変換して得られる１個のＡＮＤゲート及び１個以上のレシーバ(入力ピン)である。ドライバがイネーブルされない場合にゼロとなっている“フローティングロー”回路網を、１個以上の加算ＯＲゲートをＡＮＤゲートにより駆動することで実現する。“フローティングハイ”ゲートは、ＡＮＤゲートに反転データ入力を有しており、最終的な加算ゲートをＮＯＲとしている。同じ接続形態及び基本的な方法を、両ケースに適用する。一般的な一方向性接続及び１以上の加算

ORゲートを用いて、積の和としてトライステート回路網を具体化する。ドライバのパスを、相互接続階層のXからZに向かって集中させ、ドライバを加算ORゲートに集める。最も高いレベルの加算ORゲート出力を、ロジック回路網の真の値、すなわち、そのソースとしている。ソースを相互接続階層の下方に向かって接続し、すべてのドライバを駆動する。結果的に、幾つかのチップ対（Z－X、X－Y及び／又はY－Z）は、2つのパスを必要としている。そのうち、一方のパスはドライバを加算ORゲートとつないでいるものであり、レシーバと出力とをつなぐものである。図48は、トライステート回路網の相互接続を示しており、詳細については以下に説明する。

【0074】2.4.2 ネーミング

固有のネームを有する回路網を用いて、論理チップ内の相互接続をネットリストファイル中に規定する。これらの回路網は、設計部データ構造中の回路網と混同すべきではない。各設計部回路網は、論理ロジックチップネットリストファイル中に片方の回路網を有し、入力設計ファイルに用いたのと同じ実際の回路網ネームをネットリストファイルに用いる。基本要素変換の間に設計部データ構造に加えられる回路網に、人工的に発生させられたネームを付ける。設計部データ構造中に存在しない回路網を、ロジックチップ及びクロスバーチップネットリストファイルに送出し、相互接続を特定する。ロジックチップ又はクロスバーチップ I／OバッファとI／Oピンとの間の回路網、ANDゲートとトライステートの積の和である加算ゲートとの間の回路網、及び、クロスバー加算を用いる場合に相互接続の上方及び下方に通っている回路網、これらすべての回路網は設計部の単一回路網と関連しているが、ネットリストファイル中の回路網とは別個のものである。相互接続基本要素をネットリストファイルに送出する際は、実際の回路網ネームを変更し、これら相互接続機能の各々に対して別個の回路網ネームを提供する。次のチャートはネーム変更のすべてをリストしている。I／Oバッファとそのピンとの間のチップレベル毎に、1個のネームのみを使用する。これらのネームには、接続の他端におけるチップに従って番号を付して独自性を持たせている。チップレベル毎に1回よりも多く使用されるネームによって、クロスバーチップ内部接続を規定する。これは、このような多くの構成し得るネーミングシステムの一例にすぎない。文字'H'を、チャート中の実際の回路網ネームの代わりに用いる。例えば、相互接続された回路網を'ENABLE'と称する場合、ロジックチップ6から受信する入力バッファ入力端子とそのI／Oピンとの間の回路網を'ENABLE－D－6'と称する。

'N'：

Lチップ：このLチップを回路網のソースとする場合、真の回路網値である。このLチップに唯一つのドライバを設ける場合トライステートドライバである。

X，Y，Zチップ：1個のチャイルド（child）ドライバを設ける場合、チャイルドからの入力バッファ出力ピンである。このチップを回路網のソースとする場合、チャイルドへの出力バッファ入力ピンである。

すべてのチップ：親への出力バッファ入力ピンである。

加算ゲート出力端子。

'N_R'：

Lチップ：この回路網のソースをいずれかに設ける場合、真の回路網値である。

X，Y，Zチップ：このチップが回路網のソースではない場合、チャイルドの出力バッファの入力ピンである。

すべてのチップ：親からの入力バッファ出力ピン。

'N_R_c'：

X，Y，Zチップ：チャイルドへの出力バッファ出力ピンである。ここで'c'は、チャイルドのチップナンバである。

'N_P'：

すべてのチップ：親からの入力バッファの入力ピン。

'N_D'：

すべてのチップ：親への出力バッファの出力ピン。

'N_D_c'：

X，Y，Zチップ：チャイルドからの出力バッファの出力ピン。ここで'c'はチャイルドのチップナンバである。

'N－P'：すべてのチップ

親からの入力バッファの入力ピン。

'N－D'：すべてのチップ

親への出力バッファの出力ピン。

'N－D－c'：X，Y，Zチップ

チャイルドからの入力バッファの入力ピン。

'N_OR_i'：

Lチップ：1より多くのドライバをLチップに設けた場合、トライステートドライバである。ここで 'i'は多くのこのようなドライバを識別している。

X，Y，Zチップ：1より多くのチャイルドドライバを設ける場合、チャイルドドライバからの入力バッファ出力ピン。

すべてのチップ：加算ゲート入力端子。

【0075】2.4.3 ステージ1：ロジック基本要素のネットリスティング

ステートメントは、各基本要素毎に、設計部データ構造中のすべてのロジック基本要素に対するロジックチップネットリストファイルに送出される。基本要素を接続している回路網のネーミングを行い、以下のステージ3dの相互接続バッファに用いられるネーミングと一致させる。回路網のソースを同一の論理チップ中に設ける場合、入力ピンをこれらの本来の回路網ネームに接続する。このことは、閉回路網（カットがない回路網）に対して常に真であり、カット回路網のLチップを駆動する

際においても真である。Lチップをソースとしない場合、入力ピンをこれらの親レシーバの入力バッファに接続する。出力ピンをロジックチップの加算ゲートに接続する場合を除き、出力ピンをこれらの本来の回路網ネームに接続する。出力ピンをロジックチップの加算ゲートに接続する場合、特定の回路網ネームを変更させる。

【0076】2.4.4　ステージ2：ロジックチップ加算ゲートのネットリスティング

完全に単一ロジックチップ中に含まれているトライステート回路網の加算ゲートに関するステートメントを、各回路網毎に送出する。上述した回路網ネームの変更を用いて入力端子を接続する。出力端子は本来の回路網ネームを駆動する。回路網が"フローティングハイ"であるかどうかに応じて、適切な出力検知（OR又はNOR）を用いる。

【0077】2.4.5　ステージ3：カット回路網相互接続の決定及びネットリスティング

1より多くの論理チップ間を通っている回路網（カット回路網）の相互接続をネットリストする。カット回路網を、各々、ステージ3a、3b及び3cを介して一度に処理する。

【0078】2.4.5.1　ステージ3a：相互接続トリーの構成

一時的なトリーデータ構造を構成し、相互接続プロセスを案内する。この一時的トリーデータ構造は、この回路網に基本要素を有するLチップと、相互接続を具体化するX、Y及びZチップと、各々の相互接続の要件を示すことによって、回路網の構造を表現する。各トリーレベルでの各ノードはシステム中のロジック又はクロスバーチップに対応しており、その下位のチャイルドノードにつながっているブランチを有し、ノード及び親のパスにつながっている相互接続パスに関するデータを以下のように記憶する：

レベル	チップ	相互接続パス
ルート	Zチップ	なし
第1レベル	Yチップ	Y－Zパス
第2レベル	Xチップ	X－Yパス
第3レベル	Lチップ	L－Xパス

回路網中に含まれている各Lチップは、回路網にいかに多くの基本要素を有していても、トリー中の唯一個のノードで表現される。各ノードは以下のエントリを有している。

チップナンバ：ボードのいずれかのLチップ、ボックスのいずれかのボード又はラックのいずれかのボックス。初期値はNULL。

D及びRカウント：このノードのパスに必要とされるドライバ（D）及びレシーバ（R）の数である。初期値はゼロ。

D及びRパス：（各L－X、X－Y又はY－Zパスで利用できるいくつかのパスナンバの中から）いずれかのパスナンバを用いて、ドライバがこのノードからトリーを上り、レシーバが降りる。初期値はNULLである。

トップサム：このノードが下位にすべてのドライバを具える加算ゲートを有している場合、真と認識する。これを用いて、多数ゲートの積の和における最終ゲートを制御し、"フローティングハイ"の場合その出力反転を得る。初期値は偽である。

回路網が多数のボックスに及んでいない場合、ルートノード（root node）はナルエントリ及び唯一つの第1レベルノードを有している。回路網が多数ボードに及んでいない場合、第1レベルノードはナルエントリ及び唯一つの第2レベルノードを有している。回路網が多数のLチップに及んでいない場合、回路網は、相互接続の必要がなく、トリーを有していない。パーティショナによって割当てられた基本要素のロケーションに従って、設計部データ構造中の回路網を操作しトリーを構成する。回路網が1より多くのボックス又はボードに及んでいない場合、必要とされないクロスバーレベルのノードをナルとする。このようにして、各Lチップの駆動出力端子及び受信入力端子の数をカウントするとともにLチップノード中に記憶し、Lチップの相互接続の必要性を確認する。各Xチップノードにおいて、ドライバを有しているLチップの数及びレシーバを有している数をカウントし、各Xチップが、どのような相互接続を設けなければならいかを確認する。同様に、各Yチップにおいて、駆動及び受信Xチップをカウントし、Zチップにおいて、Yチップをカウントする。最後に、トリーを分析し、ソースである回路網の真の値をレシーバに伝送するポイントを決定する。簡単な回路網では1個のLチップの中にソースを設けている。クロスバー加算を用いているため、ソースをトライステート回路網のクロスバーチップとすることもできる。通常、クロスバーチップがチャイルドチップ間にレシーバを有している場合、クロスバーチップをネットリストし、真の値を、そのより高レベルのペアレントチップから送信する。しかし、階層中において、親チップより下位のチップがソースを有している場合、親チップは、真の値を親チップ自体又はそれより下位のチップから得る。このようにするために、クロスバーノードを走査し、ノード又はノードの派生がソースの場合にはレシーバカウントをゼロにセットする。

【0079】2.4.5.2　ステージ3b：各セットの相互接続能力の決定

各Zチップが各ボックス中の同一のYチップを接続し、各Yチップが各ボードの同一のXチップを接続しているので、相互接続されたX、Y及びZチップによって一組を形成する。リアライザシステムの好適例においては、64組を設けている。その各々は、1個のZチップと、各ボックス中に1個設けている、Zチップを用いたY－Zパスを有する8個のYチップと、各ボードに1個設けている、各Yチップを用いたX－Yパスを有する64個

のＸチップとを具えている。各々の組の対は、この場合、同一のＸチップを有しているが、このことは許容されている。その理由は、唯１個の組のみを選択し、回路網を相互接続するからである。ＬチップとＸチップのような相互接続されたチップの対の各々を、パスと称する一群のワイヤで接続する。各クロスバー中のパスを、パステーブルにリストする。Ｌ－Ｘパステーブルは、システム全体中の各Ｌ－Ｘクロスバーの各パスに関連するエレメントを有している。各ボックス中の各ボードにはＬ－Ｘクロスバーを設け、各クロスバーには、一組の各Ｌチップ及びＸチップに関係するパスを設ける。このようにして、Ｌ－Ｘパステーブルは５個の寸法を有している。すなわち、ＬＸ〔ボックス〕〔ボード〕〔Ｌチップ〕〔Ｘチップ〕〔パス〕である。同様にして、Ｘ－Ｙパステーブル、すなわちＸＹ〔ボックス〕〔ボード〕〔Ｙチップ〕〔パス〕と、Ｙ－Ｚパステーブル、すなわちＹＺ〔ボックス〕〔Ｚチップ〕〔パス〕とを設ける。テーブル中の各エレメントを、相互接続手続によって、“フリー（free）”又は“ユーズド（used）”とする。ネットリストファイルに送出された入力又は出力Ｉ／Ｏピンがパスを使用する場合、テーブルエレメシトを使用する。各組の回路網相互接続能力を、相互接続すべき各パスに対するフリーなパスカウントを捕捉することによって決定する。まず第１番目に、ボックス中のＹチップとＺチップとの間のＹ－Ｚパスについて考える。回路網中の各ボックスにおいて、この組中のＺチップ及びこのボックスのＹチップに関するＹ－Ｚパステーブル中のフリーパスの数をカウントし記憶する。第２番目に、ボード上のＸチップとボックス中のＹチップとの間のＸ－Ｙパスについて考える。すなわち、回路網中の各ボードにおいて、この組のこのボックスのＹチップ及びこのボードのＸチップに関するＸ－Ｙパステーブル中のフリーパスの数をカウントし記憶する。第３番目に、ボード上のＬチップ及びＸチップ間のＬ－Ｘパスについて考える。すなわち、回路網の各ロジックチップにおいて、この組のこのＬチップ及びこのボードのＸチップに関するＬ－Ｘパス中のフリーパスの数をカウントし記憶する。いかなるポイントにおいても、相互接続を完成するのに十分なフリーパスが存在しない場合、この組を故障と認識し、このプロセスを次の組に進める。結果的には、相互接続中の各パスすなわち首尾よく相互接続を達成することのできるクロスバーチップの各組に関するパスカウントを捕捉することとなる。

【００８０】２.４.５.３　ステージ３ｃ：組の選択

多くの組を用いて相互接続することができるので、組の１個を選択し使用するパスのバランスを保持する。このことによって、完全な相互接続機能の開発が保証される。簡単な組選択技術は、全パスカウントが最大である組を選択することである。しかし、このことは局所的な条件を無視している。すべてのレベルにおけるパスカウントの中から、最も大きな最小パスカウントを有する組を選択するのが好ましい。例えば、２組が以下のパスカウントを有しているものと仮定すると、

パス：	ＹＺ	ＹＺ	ＸＹ	ＸＹ	ＬＸ	ＬＸ	ＬＸ
組Ａ：	４	４	４	３	１	３	４
組Ｂ：	３	３	３	３	３	３	３

組Ａは、最大トータル（２３対２１）を有しているが、これを選択するということは、最後に利用することのできる、１個のＬチップ－Ｘチップ対からのＬ－Ｘパスを採用することを意味している。組Ｂは、最も大きな最小（３対１）を有しており、Ｌチップ－Ｘチップ対を閉じることはない。結合の場合、検討の結果、各組から１個の最小を排除し、組を１個選択するまで、最も大きな最小を有している組を選択する。（第１回路網の場合と）実際にすべての組が同一である場合、同一である場合、一つを採用する。これが、使用されている方法である。一組のトライステート回路網について検討を加える場合、特に考慮を要する。一方が階層の上方に向かって加算ゲートにつながっている入力端子のためのパスである。同一の回路網に使用されるこれら２個のパスを、いくつかのチップ対は、有していなければならないため、これらの場合に、選択された組は、少なくとも２個のフリーパスを有していなければならない。パスのトリーノード（すなわち、Ｌ－Ｘパス等のためのＸチップノード）がノンゼロのＤ及びＲカウントと、ノンＮＵＬＬの親とを有している場合、このような場合を検出する。

【００８１】２.４.５.４　ステージ３ｄ：相互接続のネットリスティング

組の選択及びトリー構造を与え、バッファ及び加算ゲートのステートメントをロジック及びクロスバーチップネットリストファイルへ送出することによって、相互接続をネットリストする。このことを、各レベル毎に、まずロジックチップについて行い、その後、Ｘ、Ｙ及びＺチップについて行う。各チップの相互接続及び方向性を、トリー中のデータを使用することによって決定する。相互接続のバッファ及び回路網に関してのステートメントをネットリストファイルに送出することによって、各接続をネットリストする。（チャイルドチップが存在する場合）チップとチャイルドチップとの接続を、まずネットリストする。各チャイルドチップを順番に検討する。トリーがこのチップを駆動させていることを示している場合、チャイルドチップのドライバを接続しているピンナンバを用い、入力バッファをネットリストする。このチップが１個より多くのドライバを有している場合、別の回路網ネームを各々に用いる。このようにして、これらの回路網ネームは後にネットリストされる加算ゲートによって捕捉される。チャイルドがこのチップを受信していることをツリーが示している場合、チャイルドチップのレシーバを接続しているピンナンバを用いて、出力バッファをネットリストする。このチップ自体がその親

からのレシーバである場合、異なる回路網ネームを用い、このチップは親レシーバを接続する。このチップが、そのチャイルド中に1より多くのドライバを具えている場合、加算ゲートをネットリストし、上述したドライバ回路網を接続する。最終的に、(存在するならば)親チップへの接続をネットリストする。チップ又は任意の派生チップがドライバを具えている場合、チップの対及び選択された組に関するパステーブルエントリからドライバに関する相互接続パスを採用するとともに出力バッファをネットリストし、ここで採用したパスを介して親を駆動する。このチップが、ペアレントからのレシーバである場合、パステーブルからパスを選択し、このパスを用いて入力バッファをネットリストする。

【0082】2.4.6 相互接続及びネットリスティング手続についての詳細な規定:

第1の一般的な規定:

回路網に関して4個のクラスを設ける:

シンプルな閉回路網:回路網は1個のドライバを有し、すべての基本要素を同一のLチップ中に設ける。

シンプルなカット回路網:回路網は1個のドライバを有し、基本要素を多数のLチップ中に設ける。

トライステート閉回路網:回路網が1個より多くのドライバを有し、すべての基本要素を同一のLチップ中に設ける。

トライステートカット回路網:回路網が1個より多くのドライバを有し、基本要素を多数のLチップ中に設ける。

回路網の'ソース'とは、その実際の論理値を伝送するチップである:シンプルな回路網では、ソースはドライバを有しているLチップである。トライステート回路網では、ソースはトップモスト(top-most)加算ゲートを有しているチップである。

これを決定するために:回路網を走査し、どこに出力ピンを配置しているかを調べる。出力ピンがすべて同一のLチップ上に存在する場合、このLチップがソースである。これ以外の場合で、出力ピンがすべて同一のボード上に存在する場合、このボード上のXチップがソースである。これ以外の場合で、出力ピンがすべて同一のボックス中に存在する場合、このボックス中のYチップがソースである。上記以外の場合、Zチップがソースである。出力ピンのインデックスナンバは、それが、その回路網におけるピンの循環リストにおけるどの出力ピンであるかを示しており、ネットレコードが示すピンから始まり、ゼロから一つずつカウントする。

ステージ1:設計部データ構造中のすべての基本要素を送出する。

設計部データ構造中の各Lチップは:Lチップのネットリストファイルがオープンされていない場合には、このLチップのネットリストファイルをオープンする。

このLチップの各基本要素は:基本要素へッダステートメントをファイルに送出する。

この基本要素の各ピンは:(入力設計ファイルからネームを得るための回路網の対象識別子を用いて)接続された回路網のネームを得る。そして、'N'と称する。

入力ピンの場合:このLチップが回路網のソースを有している場合には、回路網'N'に接続された入力ピンに関するステートメントを送出する。そうでない場合には、回路網'N_R'における入力ピンステートメントを送出する。

出力ピンの場合:この出力ピンのインデックスナンバを得て、'p'と称する。シンプル回路網の場合には回路網'N'のピンに指示を出す。トライステート閉回路網の場合には回路網'N_OR_p'のピンに指示を出す。

トライステートカット回路網の場合:これがこのLチップのこの回路網に関する唯一つの出力である場合、回路網'N'のピンに指示を出す。唯一つの出力ではない場合、回路網'N-OR-p'のピンに指示を出す。

次のピンへ。次の基本要素へ。次のLチップへ。

ステージ2:すべての閉回路網加算ゲートに指示を出す:

各トライステート閉回路網は:'N'と称するこの回路網のネームを得る。このLチップのネットリストファイルがオープンされていない場合には、これをオープンする。'i'と称する回路網に、何個の出力端子が存在するかをカウントする。

'i'入力ゲートに関するステートメントを送出する:この回路網が'フローティングハイ'である場合にはNORであり、これ以外の場合にはORであり、(0～i-1のすべてのjについて)回路網'N_OR_j'に接続された入力端子と、'N'に接続された出力端子とを有している。次の回路網へ。

ステージ3:カット回路網と相互接続しているバッファに指示を出すとともに、すべてのカット回路網加算ゲートに指示を出す:すべての相互接続パステーブルのすべてのエレメントを"フリー"にする。各カット回路網(シンプル又はトライステート)は:階層の順番でカット回路網を選択し、第1ボックス回路網等を選択するとともに、この順番の範囲内で最も大きな第一番目の回路網を選択する。

ステージ3A:トリーの構成

回路網の各基本要素は:この基本要素のボックスにトリーノードを設けていない場合には、1を加える。このボックス中の基本要素のボードにトリーノードを設けていない場合には、1を加える。このボックス中のこのボード上のこの基本要素のLチップにトリーノードを設けていない場合には、1を加える。この基本要素の回路網接続が出力ピン(すなわちドライビィング(driving))である場合には、このLチップのノードのDカウントをインクリメントする。上記以外の場合で、このLチップ

がこの回路網のソースではない場合、このＬチップのノードのＲカウントをインクリメントする。次の基本要素へ。この回路網のすべての基本要素をトリーで表現する際、唯一つのＸチップノードのみを設けるならば、ＹチップノードをＮＵＬＬとする。（すなわち、回路網はボード上に存在している。）唯一つのＹチップノードのみが存在する場合、ＺチップノードをＮＵＬＬとする。（回路網はボックス中に存在する。）
各ノンＮＵＬＬクロスバーレベルにおいて、まずＸチップ、その後Ｙチップ、その後Ｚチップ：このレベルにおける各ノードは：
Ｄ＝〔Ｄカウントがゼロでないチャイルドノードの数〕
Ｒ＝〔Ｒカウントがゼロでないチャイルドノードの数〕
このノード又は派生をこの回路網のソースとする場合、このノードをＲ＝０にセットする。このノードをソースとし且つ回路網をトライステートにする場合、その“トップサム（top sum）”フラグを真にセットする。次のノードへ。次のレベルへ。
ステージ３Ｂ：各組の相互接続能力の決定
各組は、相互接続すべき各パスのパスカウントを捕捉し、その相互接続能力を決定する。この組のパスカウントの記憶を割当てる：
Ｙ－Ｚパスカウント：各ボックスの割当て
Ｘ－Ｙパスカウント：各ボードの割当て
Ｌ－Ｘパスカウント：各Ｌチップの割当て
唯一個のボックスのみをこの回路網中に設ける場合：このボックスのナル（ゼロではない）Ｙ－Ｚパスカウントをそのままにしておく。
そうでない場合には：各ボックスは：パスアレイ中のフリーパスの数をカウントする。
ＹＺ〔このボックス〕〔この組〕〔パス〕
このボックスのトリーノードがノンＮＵＬＬの親を有し、且つＤ＞０及びＲ＞０である場合、このボックスのパスは“ダブル”である。すなわち、ドライバとレシーバの両方を有している。フリーパスが２よりも少ない場合、この組はこの回路網を接続できない。以上以外の場合で、且つパスが存在しない場合、この組はこの回路網を接続できない。この組が接続できない場合、これを使用不可能とし、次の組に進める。接続できる場合には、トータルを、このボックスのＹ－Ｚパスカウントとしてセーブする。この回路網に、唯一のボードのみを設けている場合、（Ｙ－Ｚ相互接続は必要とされない）：このボードのナル（ゼロでない）Ｘ－Ｙパスカウントをそのままにしておく。２以上のボードを設けている場合：各ボードは：パスアレイなかのフリーパスの数をカウントする。
ＸＹ〔このボックス〕〔このボード〕〔このセット〕〔パス〕
このパスが“ダブル”であり且つパスが２より少ない場合、又は、パスを設けていない場合、この組はこの回路網を接続できない。この組を使用不可能とし、次の組に進む。上記でない場合：トータルを、このボードのＸ－Ｙパスカウントとしてセーブする。各Ｌチップは：パスアレイ中のフリーパスの数をカウントする。
ＬＸ〔このボックス〕〔このボード〕〔このＬチップ〕〔この組〕〔パス〕
このパスが“ダブル”であり、且つパスが２より少ない場合、又は、パスを設けていない場合、この組は、この回路網を接続できない：この組を、使用不可能とし、次の組に進む。上記でない場合には、トータルを、このＬチップのＬ－Ｘパスカウントとしてセーブする。このボードの次のＬチップへ。このボックスの次のボードへ。次のボックスへ次の組へ
ステージ３Ｃ：回路網の選択：回路網を接続することのできる各組は：この組のすべてのパスカウントの中から最小パスカウントを見出す。次の組へ。これらの最小パスカウントの中から最大のものを見出す。検討の後、最大の最小パスカウントよりも少ないパスカウントを有するすべての組を除去する。組が存在しない場合には、この回路網を相互接続することができない。一組だけが残る場合には、この回路網の組を選択する。２以上の組が残る場合には、すべての最小パスカウントの中から次の最も大きい最小を見出す。検討の後、これよりも小さいパスカウントの組すべてを除去する。一組が残るまで、又は、すべての残っている組のパスカウントが同一となるまで、これを繰り返す。この回路網の残っている組の内の任意の１個を選択する。すべての組のすべてのパスカウントの記憶を解除する。ステージ３Ｄ：相互接続のネットリスト以下で用いられる手続の規定：ドライバ（又はレシーバ）パスを得る又は確保するために：
１）このレベルのパステーブル中のフリーエレメント、このノードのチップナンバ及びペアレントノードのチップナンバから、パスを選択する。
２）使用されたパスのテーブルエレメントをマークする。
３）このノードのドライバ（又はレシーバ）パスナンバエントリの中のパスナンバとして、どのパスを用いたかを記憶する。Ｉ／Ｏピンナンバを導出するために：
１）２個のノードのチップナンバ及び組ナンバから、このノードのチップとチャイルドノードのチップ（又は、場合によっては親ノードのチップ）との同一性を確認する。これによって、含まれている特定のパス（例えば、Ｌ４－Ｘ５，又はBoard３－Ｙ７のようなパス）を識別する。
２）パスナンバがチップの対を接続している幾つかのパスの内の一つのパスを示しているということをリコールする。チップ、パス及びパスナンバを与え、Ｉ／Ｏピンナンバ情報を有している索引テーブルから、このパスを接続しているピンナンバを読出す。パスを用いて、バッファ（入出力）に指示を出すために：

1）パスナンバをこのノードから得る。又は、チャイルドパスが特定される場合には、パスナンバをそのチャイルドノードから得る。ドライバ又はレシーバパスナンバを、指示されたようにして得る。
2）パスナンバを用いて、このバッファのＩ／Ｏピンナンバを得る。
3）入力バッファであるか出力バッファであるかに応じて、このノードチップのネットリストファイルに、基本要素ステートメントを送出する。この際、指示されるように、入出力回路網ネームを用いるとともに、そのＩ／Ｏピンに対して得られたピンナンバを用いる。
相互接続のネットリスト手続：‘Ｎ’と称するこの回路網のネームを得る。各ノンＮＵＬＬレベルの第１Ｌチップ、その後Ｘチップ、Ｙチップ及びＺチップにおいて：トリー全体におけるこのレベルにおける各ノードは：準備していない場合には、このノードのチップに関するネットリストファイルをオープンする。レベルがＸ、Ｙ又はＺである場合：このノードの下位の各チャイルドノードは：カウンタ‘ｉ’をゼロにセットする。チャイルドのＤ＞０である場合：（チャイルドとはドライバである）このノードのＤ＝１である場合：‘Ｎ＿Ｄ＿ｃ‘から‘Ｎ’へと入力バッファに指示を出す。（ここで‘ｃ’は、チャイルドノードのナンバである。）この際チャイルドドライバパスを使用している。このノードのＤ＞１である場合：‘Ｎ＿Ｄ＿ｃ’から‘Ｎ＿ＯＲ＿ｉ’へと入力バッファへと指示を出す。この際、チャイルドのドライバスを使用し、‘ｉ’をインクリメントする。チャイルドのＲ＞０である場合：（チャイルドはレシーバである）このノードのＤ＞０、且つこのノードのＲ＝０の場合：‘Ｎ’から‘Ｎ＿Ｒ＿ｃ’へと出力バッファに指示を出す。この際、チャイルドシートのレシーバパスを使用する。そうでない場合：‘Ｎ＿Ｒ’から‘Ｎ＿Ｒ＿ｃ’へと出力バッファに指示を出す。この際、チャイルドのレシーバパスを使用する。次のチャイルドノードへ。このノードのＤ＞１である場合：（ノードは加算ゲートを有している）‘ｉ’入力ゲートに指示を出す：この回路網が‘フローティングハイ’であり、且つこのノードの‘トップサム’フラグが真である場合、ＮＯＲである。そうでない場合には、ＯＲである。この際、（０～ｉ－１のすべてのｊに対する）‘Ｎ＿ＯＲ＿ｊ’に接続された入力端子と、‘Ｎ’に接続された出力端子とを有している。このノードのＤ＞０で且つこのノードがノンＮＵＬＬ親を有している場合：（ノードはドライバである）ドライバパスを確保するとともに、受信する。‘Ｎ’から‘Ｎ＿Ｄ’へと出力バッファに指示を出す。この際、ドライバパスを使用する。このノードのＲ＞０である場合：（ノードはレシーバである）レシーバパスを得て確保するとともに、受信する。‘Ｎ＿Ｐ’から‘Ｎ＿Ｒ’へと入力バッファへと指示を出す。この際、受信パスを使用する。このレベルの次のノードへ。次のレベルへ。次のカット回路網へ。すべてのオープンネットリストファイルをクローズする。

【００８３】２.４.７　２例の回路網
図４７ａは、‘ＢＸ’と称する、シンプル回路網のオリジナル入力設計部を示している。これは、１個のドライバと３個のレシーバとを具え、同一ボックス中の一方のボード上の２個のロジックチップ及び他方のボード上の１個のロジックチップに及んでいる。この回路網のステージ３ａによって構成された相互接続トリーを、図４７ｂに示す。どのようにして、各ロジックチップ、各ボードに対するノード及びボックスに対するノードを設けるかに注意しなければならない。ロジックチップノードは、特定のロジックチップに対応している。ボードノードは各ボード上のＸチップに対応しており、ボックスノードはＹチップに対応している。Ｚチップは、この回路網では必要ない。正確には、どのＸ及びＹチップを使用するかはどの組を選択するかに依存しており、これは、トリー中には示されていない。Ｄ及びＲの値を各ノード毎に示している。ノードがレシーバであっても、Ｌ０がＤ＝０であることに注意する。その理由は、ノードがこの回路網のソースノードであり、他のノードとは異なりソースノードから値を受信する必要がないからである。ボード２のノードにおいては、そのＲカウントは初期値が１であり、Ｌ４のレシーバをカウントする。ソースが派生であるため、Ｒカウントがゼロにセットされていたことを示している。送出された回路網ネームは、これらの回路網を用いて示されている。実際の相互接続の構造によって、トリーの構造及び各ノードのＤ及びＲカウントをどのようにして表わすかを述べる。図４８ａは、‘ＥＸ’と称するトライステート回路網のオリジナル入力設計部を示している。これは、同一ボックス中の一方のボード上の２個のロジックチップ及び他方のボード上の１個のロジックチップに及んでいる３個のトライステートドライバと、２個のボックスにおける３個のボード上の４個のＬチップに及んでいる６個のレシーバとを具えている。この回路網のステージ３ａによって構成される相互接続ツリーを、図４８ｂに示す。この回路網はボックスに及んでいるため、Ｚレベルクロスバーを用いる。２個のトライステートドライバを有しているために、ボード２のノードはＤ＝２であることに注意しなければならない。Ｘチップは、加算ゲートを具え、ボード２のＬチップからのタームを捕捉する。回路網のソースであるボックス２のノードも同様であり、これを、“トップサム”とする。このＹチップは、トップモスト加算ゲートを具え、ボード２及び３からのタームを捕捉する。ボックス２のノード及びそのＺペアレントノードはソースを具え、これらのＲカウントをゼロにする。各ロジックチップ及びクロスバーチップに関するネットリストファイルに送出される実際のゲート及びバッファと、いかにして、これらを相互接続するかを、図４９に示

す。設計変換によって、各トライステートドライバをどのようにＡＮＤゲートに変換するかを注意しなければならない。これらの出力を、Ｘ及びＹレベルの加算ゲートによって捕捉する。受信入力は、“トップサム”ノード、すなわちボックス２のＹチップから伝送される。ボックス２のレシーバは、相互接続へつながっているパスによって駆動される。ボックス６のレシーバは、Ｚレベルクロスバーチップを介して駆動される。

【００８４】３　リアライザシステムの応用

３．１　リアライザロジックシミュレーションシステム

ロジックシミュレータは、ハードウェア又はソフトウェアによって実現されるシステムである。このシステムは、入力設計、一組の設計部への刺激、及び或る期間中の刺激の方向を受信するとともに、一組の刺激を出力する。この刺激によって、実際の入力設計部を実現し、所定の同一の刺激を発生させることを予測する。刺激及び応答は、特定の時間に、特定の設計回路網のロジック状態を伝送するものである。シミュレータユーザが、入力設計ファイルの形態で設計部の記述のみを供給するということが重要な特性であり、短期間に設計部を変更するとともに、これに再度刺激を与えることができる。現在のソストウェアロジックシミュレータ設計部の演算は、コンピュータソフトウェアプログラムを用いており、設計部のオペレーションを予測するシーケンシャルなアルゴリズムを実行する（“An Introduction to Degital Simulation”,Mentor Graphics Corp.,Beaverton,Oregon,1989）。よく知られているように、イベントドライブされたコードアルゴリズム、又は、コンパイルされたコードアルゴリズムのいずれか一方を使用する。現在のハードウェアロジックシミュレータ設計部の演算とは、ソフトウェアシミュレータに使用されるのと同一の、イベントドライブされたコードアルゴリズム、又は、コンパイルされたコードシーケンシャルアルゴリズムを実行するハードウェアを構成することである。アルゴリズムの並列処理を開発及び／又は特別なアルゴリズムオペレーションを直接実現することで、ハードウェアはその実行による利益を得ることができる。このことは、一般的な目的のコンピュータ実行ソフトウェアにおいては不可能である。現在のハードウェアロジックシミュレータは、入力設計部の応答を予測するシーケンシャルなアルゴリズムを実行することで動作する。ロジックシミュレータを構成する新しい手段は、リアライザシステムに基づいている。リアライザロジックシミュレータシステムは、入力設計を受信し、これをリアライザハードウェアのロジック及び相互接続チップの構成に変換する。この際、リアライザ設計変換システムを使用する。リアライザロジックシミュレータシステムは、一組の設計部への刺激と、ある期間のシミュレートする方向とを受信し、ベクトルメモリを介し、実現される設計部に刺激を与え、ベクトルメモリを介して、実現される設計部からの一組の応答を捕捉する。応答は、入力設計部を実際に実現することによって、所定の同一の刺激を発生させることに対応している。その理由は、前記刺激に対応させて、設計部をハードウェアによって実際に実現するからである。このことは、現在のロジックシミュレーションシステムのすべてが、設計部の刺激に対する応答を予測するシーケンシャルアルゴリズムを実行するが、リアライザロジックシミュレータは実際の設計部の実現を行い、設計部の刺激に対する応答を決定するという点において、すべての現行のロジックシミュレーションシステムとは異なる。主な利点は、実現された設計部が、シーケンシャルアルゴリズムが応答を予測できるより速く、種々の速さで応答を発生させるということである。リアライザロジックシミュレーションシステムは、（すでに説明した）リアライザ設計変換システムと、ロジックシミュレータ刺激及び応答伝送システムと、リアライザハードウェアシステム及びホストコンピュータと相俟って、カーネルを作動させるロジックシミュレータとから成っている（図５０）。

【００８５】３．１．１　ロジックシミュレーション刺激及び応答の変換システム

このシステムは、ユーザが作成した刺激イベント入力ファイルを、直接ベクトルメモリにロードすることのできる刺激データを含むバイナリーファイルに変換するとともに、ベクトルメモリから読出されるバイナリー応答データを有するファイルから、ユーザが読出し可能な応答イベント出力ファイルへ、応答を変換する。刺激及び応答イベントは、回路網ネーム、時間及び新しい回路網の状態値から成っている。変換とは、回路網ネームとベクトルメモリビットとの間の変換、及び、シミュレーションの‘実時間’とベクトルメモリロケーションとの間の変換である。時間変換は、刺激イベントを有する各特定の時間をベクトルメモリロケーションに印すとともに、このベクトルメモリロケーションにおける応答イベントをこの時刻に発生したものとして報告する。好適例では、刺激入力イベントファイル及び応答出力イベントファイルを、Mentor Graphics Logfiles（“Quick Sim Family Reference Manual”, MentorGraphics Corp.,Beaverton,Oregon,1989）としている。これは、一連の時刻、回路網ネーム及び、新たな回路網状態値を含むテキストファイルである。ＥＤＡシステム中のバッチシミュレーションインタフェースツールによって、刺激入力イベントファイルを作成するとともに、応答出力イベントファイルを解釈する。好適例では、このツールを、Mentor Graphics のRSIMツールとする。ここでは、このセクションで後に説明するように、すべての基本要素を、ゼロディレイでシミュレートするものと仮定する。刺激イベント入力ファイルを、刺激バイナリーファイルへ変換するためには：

１）刺激入力イベントファイルを読出す。時間の経過に

応じて、刺激イベントをオーダするとともに、何個の異なる時刻がイベントを有するかを決定する。
2）設計変換システムが出力するこの設計部中の各ベクトルメモリに対する対応テーブルを読出す。
3）各ベクトルメモリロケーションは、1以上の刺激イベントを有する時刻に対応している。各々の異なる刺激イベント時刻に、十分なベクトルメモリロケーションが存在しない場合、ステップ5及び6を必要なだけリピートし、すべてのこのような時刻に十分な刺激バイナリーファイルを出力する。このファイルは、各々メモリに適合する刺激を有している。
4）ベクトルアレイ“V0”，“V1”等の記憶を割当てる。その各々は、ロケーションのナンバ及び回路網幅と一致しており、シミュレートされる設計部に用いられるベクトルメモリを用いている。ベクトルアレイと同じ長さを有するタイムアレイ“T”の記憶を割当てる。
“ラストベクトル”バッファ“B0”、“B1”等を割当てる。このバッファは、各ベクトルメモリに対するものであり、各々その回路網と同じ幅であり、これらをゼロに初期化する。
5）ベクトルアレイインデックスカウンタ‘v’を、ゼロにセットする。1以上の刺激イベントを有する各時刻のうち、最も早い第1番目の時刻において、ベクトルメモリ‘n’及び、この回路網のベクトルメモリビットポジション‘i’を設定する。この際、このイベントの回路網の対応テーブルエントリを使用する。このイベントに対する新しい値を、Vn[v] ビットi及びBnビットiに書込む。次のイベントへ。V0[v]、V1[v] 等の内容の各々を、B0、B1等に書込む。T[v]中のこの時刻を記憶する。vをインクリメントする。刺激イベントを有する次の時刻へ。
6）ベクトルアレイV0、V1等、タイムアレイT及びサイクルカウント‘v’を刺激バイナリーファイルに書込む。応答バイナリーファイルを、応答イベント出力ファイルに変換するために：
1）ベクトルアレイV0、V1等、タイムアレイT及びサイクルカウント‘v’を応答バイナリーファイルから読出す。各ベクトルメモリロケーションは、1以上の刺激イベントを有する時刻と一致している。各々異なる刺激イベント時刻に対して、ベクトルメモリロケーションが十分でない場合、ステップ1～4を必要なだけ繰り返し、すべての応答バイナリーファイルをこれらのアレイ中に読出す。
2）設計部変換システムが出力する、この設計部中の各ベクトルメモリに対する対応テーブルを読出す。
3）“ラストベクトル”バッファ“B0”、“B1”等を割当てる。これは、各々のベクトルメモリに対するものであり、各々その回路網と同じ幅を有しており、これをゼロに初期化する。
4）ベクトルアレイインデックスカウンタ‘v’をゼロにセットする。ベクトルアレイ中の各ロケーションは：V0[v]をB0と比較し、V1[v]をB1と比較する。Vn[v]のビットとBnとの各々の差によって：このビットのベクトルメモリ及びベクトルメモリビットポジションに対応する回路網ネームを配置する。この際、このメモリに対する対応テーブルを使用する。新しい応答イベントを出力ファイルに書込む。この際、回路網ネーム、新しいビットの値及び時刻T[v]を用いる。次のイベントへV0[v]、V1[v]等の内容の各々を、B0、B1等に書込む。vをインクリメントする。次のロケーションへ。
【0086】3.1.2　ロジックシミュレーションオペレーティングカーネル
オペレーティングカーネルは、シミュレートされる設計部のリアライザシステムを構成し、刺激を与えるとともに応答を捕捉する。このことは、ホストコンピュータが行う。各セクションに説明されているように、オペレーティングカーネルはロジックチップ及び相互接続チップを構成し、ベクトルメモリ及び設計メモリを読出し及び書込むとともに、ホストインタフェースを介してクロック発生器及びリセット発生器を制御する。シミュレーションを実行するために：
1）設計部の構成ファイルを読出すとともに、これを用いて、構成のセクションで説明したように、すべてのリアライザロジックチップ及び相互接続チップを構成する。初期設計メモリデータをファイルから読出すとともに、これを設計部メモリに書込む。
2）刺激バイナリファイルを読出す。ホストインタフェースを介して、対応するベクトルメモリ中にベクトルアレイの内容を記憶する。
3）ベクトルメモリモジュール中のすべてのベクトルメモリの内容をクリアする。設計部リセット発生器を周期化し、実現される設計部を初期化する。
4）‘v’周期のECLK回路網のクロック発生器をイネーブルする。このことによって、ベクトルメモリがこれらの刺激データを送出でき、この刺激に従って、実現される設計部を作動させるとともに、ベクトルメモリが応答データを捕捉する。このことについては、刺激／応答のセクションにおいて説明されている。
5）ベクトルメモリの内容を読出すとともに、これらをタイムアレイ“T”及びサイクルカウント“v”とともに、応答バイナリーファイルに記憶する。
6）ベクトルメモリの容量が不十分であるために1より多くの刺激バイナリーファイルを設ける場合、各ファイル毎にステップ2～5を繰り返す。
7）ユーザ試験のためのファイル中の設計メモリ内容をセーブする。
【0087】3.1.3　リアライザロジックシミュレーションシステムの使用
リアライザロジックシミュレータを用いて、入力設計部

をシミュレートするために：
１）ベクトルメモリ接続を示している特性を用いて、刺激すべき回路網とこの応答を捕捉するための回路網とをマークすることによって、ＥＤＡシステムの設計作成ツールを使用し、入力設計部を準備する。必要ならば、初期設計部メモリデータファイルを準備する。ＥＤＡシステムのバッチシミュレーションインタフェースツールを用いて、刺激イベント入力ファイルを準備する。
２）リアライザ設計変換システムを用いて入力設計部を変換し、構成ファイル及びベクトルメモリ回路網対応テーブルファイルを出力する。
３）刺激及び応答変換システムをランさせ、刺激イベント入力ファイルを刺激バイナリーファイルに変換する。
４）オペレーティングカーネルをランさせ、シミュレーションを行うとともに、応答バイナリーファイルを出力する。
５）刺激及び応答変換システムをランさせ、応答バイナリーファイルを応答イベント出力ファイルに変換する。
６）ＥＤＡシステムのバッチシミュレーションインタフェースツールを用い、応答イベント出力ファイルを翻訳する。
７）入力設計部、初期設計部メモリファイル及び／又は刺激イベント入力ファイルをシミュレーションの結果によって示されるように変更し、必要ならば、ステップ２）～６）をリピートする。
リアライザロジックシミュレーションシステムの会話形の変形例では、刺激に対してはスティミュレータを用い、応答に対してはサンプラを用いている。構成及びオペレーションは以下を除き同様のものである。すなわち、バッチシミュレーションインタフェースツールの代わりに会話形シミュレーションインタフェースツールを用いて、ファイルを介する代わりに刺激及び応答変換システムと直接通信しており、又、同時に作動する会話形シミュレーションインタフェースツールオペレーティングカーネルを用いて、刺激及び応答変換システムが、ファイルを介する代わりに直接オペレーティングカーネルと通信している。イベントを有する各タイムステップを、ベクトルメモリロケーションではなく、エッジ検知タイプのシミュレータに作成する。
【００８８】３.１.４　３以上の論理状態の実現
リアライザシステムにおいて、２個のロジック状態を実現するのは実用的なことである：高ロジック状態（Ｈ）すなわち真と、低ロジック状態（Ｌ）すなわち偽であり、リアライザシステムの単一の信号を用いて、入力設計部中に直接各回路網を実現することによって実現される。ロジックシミュレーション環境において、時には３以上のロジック信号の状態を表現することが望まれる。例えば、第３の状態、“未知の（Ｘ）”を用いて、初期化されていないロジック変数又はあいまいなロジック状態を表現する。高インピーダンス状態（Ｚ）は、トライステートバスのようなワイヤ結合のバスを実現するのに役立つ。リアライザシステムの幾つかの例において、高インピーダンス状態を直接実現することができる。例えば設計部にトライステート回路網が必要とされる場合、ロジックチップ及び任意の必要な相互接続がトライステートバス機能を構成する能力を有している限り、リアライザシステムのトライステートバスによって設計部が実現される。代わりに、１個の回路網を２以上の信号へ、以下のように符号化することによって任意のロジック状態を実現する：実現すべき状態の数を決定する。すべての状態を単一に符号化するのに必要とされる最小バイナリービット数を決定し、これを‘ｎ’と称する。‘ｎ’個の実際のバイナリーロジック信号によって設計部中の回路網を実現する。例えば、３個の状態（Ｈ，Ｌ，Ｘ）を必要とする場合、２個の実際のバイナリー信号を用いてリアライザシステム中の単一の設計回路網を実現する。基本要素変換ステージの間にこの変換を行い、これらの新しいバイナリー信号を設計部データ構造に入力し、オリジナル設計回路網を置換する。更に、設計部中のロジック基本要素を、多数状態ロジック機能に従って動作するロジック回路網で実現する。例えば３個の状態（Ｈ＝高ロジック状態，Ｌ＝低ロジック状態，Ｘ＝未知）を用いる場合、設計部中の２入力ＡＮＤゲートを、３状態ＡＮＤ機能に従って動作するロジック回路に従って実現する（図５１ａ）。いずれかの入力端子をＸとし、入力端子が存在しないことをＬとする場合、出力端子に生じるＸ状態を用いて、３状態シミュレータの如く、ロジック機能が動作する（図５１ｂ）。この回路網は、２個の２ビット入力端子と、１個の２ビット出力端子とを有している（図５１ｃ）。マルチステートを実現するこの技術を、設計解析のために必要とされるように入力設計部の全体又は設計部の一部分にのみ用いることができる。２個より多くの状態でシミュレートされる回路網をこのようにして入力設計ファイル中に作成し、設計部リーダは設計部データ構造中のこのことに注目し、基本要素コンバータは基本要素に対する上記の代用回路網及び基本要素に対する多数の回路網を作成する。ロジック基本要素が、２状態の回路網接続と３以上の状態の回路網との混合を有している場合、回路網の条件に従って作動するロジック回路網を使用する。もしそうでなければ、上述したシミュレーション動作となる。
【００８９】３.１.５　リアライザのディレイの表現
現在のロジックシミュレータでは、種々の方法で、信号がロジックエレメントを通過する際のタイムディレイを作る。リアライザのロジックチップ中のロジックは実際のハードウェアであるため、そのディレイ特性は完全に正確に規定することはできず、ロジックディレイを直接作成することはできない。ロジックディレイは、プログラムを実行するシミュレータ中の特別の方法を用いることによって、及び／又は設計変換プロセスの間、ディレ

イを形成する特別のロジック機能を挿入することによって作成する。ゼロディレイ、ユニットディレイ又はリアルディレイとして、実現するシミュレーション中にディレイを形成することができる。この選択はユーザによって成され、これをリアライザロジックシミュレータシステムに指定する。

【００９０】３.１.５.１　ゼロディレイ

ゼロディレイとは、リアルタイムディレイを形成することなく、ディレイをゼロとして処理し、シミュレーションを行うことをいう。例えば、刺激イベントが、時刻‘ｉ’に、結合ロジックのみを介して出力端子と接続されている入力端子に生じる場合、この出力端子の応答イベントは、時刻‘ｔ’に生じるものとして報告される。ゼロディレイのために、設計部変換システムは特別なロジック機能を挿入しない。上記の如く、メインリアライザロジックシミュレーションシステムにおいて説明した方法に従って、シミュレーションを行う。

【００９１】３.１.５.２　ディレイ依存機能

設計部に任意の遅延依存機能を設けると複雑なものとなる。ゼロ遅延タイミングモデルではここまでには至らない。クローズドループ機能、すなわちクロスカプルドゲートのように、非同期フィードバックを設ける場合、無条件に記憶装置を設ける。記憶機能は相対的なディレイに依存している。ディレイ依存機能は、ディレイをオープンループ機能に用いると他の形態のものとなる。この一例としては入力端子で接続されたディレイ素子を有するイクスクルーシブＯＲゲートがある（図５２ａ）。イクスクルーシブＯＲゲートの出力は、信号がディレイ素子を介して伝播するのに必要とされる時間においてはハイである。この回路網に供給される信号が変化すると出力端子にパルスを出力する（図５２ｂ）。実際のリアライザロジックディレイはゼロではないため、これを直接コントロールできない場合、クロスカプルドゲートのような多くのクローズドループ及び幾つかのオープンループの場合、ディレイ依存機能は正しく動作する。しかし、ユーザは確実に実現された設計部が意図するように動作することを要求する。現在のタイミング分析ツールは、非同期フィードバックの瞬間を自動的に見出し、レポートするとともに、オープンループディレイに依存する行動を検出する。リアライザ設計変換システムは、ユーザが必要とするならば、タイミング分析ツールを用いることによってタイミング分析を行う。好適実現例では、Mentor Graphics Quick Pathタイミング分析ツールを用いる（“Quick Path User's Manual”, Mentor Graphics Corp., Beaverton, Oregon, 1989）。

１）設計変換プロセスの一部として、ＥＲＣＧＡネットリスト変換ツールは、内部相互接続及びロジック遅延の評価を出力する。これらは、レポートファイルへ送出される。

２）すべてのネットリストを変換した後、データをリポートファイルから読出し、設計部データ構造中に入力する。この際、基本要素又は回路網と関連する各ディレイ評価を用いる。

３）設計部データ構造を設計ファイルに書込む。

４）タイミング分析ツールを設計ファイルに適用する。タイミングアナライザによって検出される任意の起こり得る変化をユーザに報告する。ユーザは、入力設計ファイルを適切に評価し、修正する。

【００９２】３.１.５.３　ユニットディレイ

ユニットディレイモデルとは、各ロジック基本要素が１単位（ユニット）のディレイを有するように形成したものである。ディレイ依存性を用いて、このような形成を、時々設計部に使用し、正しい動作を保証する。適切な特性を入力設計ファイル中の基本要素に加えることによって、ユーザはゼロディレイ基本要素と合成されたユニットディレイ基本要素を指定する。自動的に、各ユニットディレイロジックエレメントの出力端子にフリップフロップを設け、ユニットディレイを形成する。これらのフリップフロップを共通クロックに接続し、この共通クロックは、第２クロック発生器によってシミュレーションの各単位時間毎に一回の周期を成す。これらのフリップフロップ及び‘タイムクロック’回路網を、基本要素変換プロセスによって設計データ構造に加える。ユニットディレイを用いてシミュレートされるロジック設計回路網の一例としては、クロスカプルドゲートを用いて作成されたフリップフロップがある（図５３ａ）。各ゲートを、その出力端子にユニットディレイフリップフロップを設けて構成する（図５３ｂ）。連続的なタイムクロック及び入力信号を与える最終的なオペレーションは、ユニットディレイゲートを有するフリップフロップのオペレーションである（図５３ｃ）。ユニットディレイシミュレーションのためのリアライザロジックシミュレータは、以下の変更を伴うものの、ゼロディレイと同様の方法を用いて動作する。

・ユーザは、どの位の時間が１ユニットに相当するのかを指示する。

・刺激及び応答回数を、ユーザが指定する時間単位の倍数‘Ｍ’に制限する。

・各ベクトルメモリロケーションは、Ｍ時間単位に対応しており、この時の刺激イベントが存在しているかどうかとは無関係である。

・刺激及び応答変換システムは、これらの仕様を用い、これらの対応関係に従って、イベントとベクトルメモリロケーションとの間にマップを作成する。

・最終的に、刺激イベントを有していない時刻は、前のロケーションと同一の内容を有するベクトルメモリロケーションによって表現される。

・オペレーティングカーネルは、‘タイムクロック’クロック発生器の周波数をＥＣＬＫの周波数のＭ倍にセットし、互いに同期をとりながら動作するように指示す

る。オペレーションの間、各M時間単位毎に1個のＥＣＬＫ、従って一組の刺激及び応答が存在する。

【００９３】３.１.５.４　リアルディレイ

リアルディレイ、すなわち種々の時間単位によるディレイを、ロジックチップ中の特定のハードウェア構成を用いて実現する。このハードウェア構成は、設計変換の間、各リアルディレイロジックエレメントの設計部データ構造に自動的に挿入される。これには幾つかの技術がある：各ロジック基本要素出力端子と直列にシリアルシフトレジスタを構成する。その長さを、各ケースに必要とされるディレイ単位の数に対応させて構成する。すべてのシフトレジスタを、各時間単位に対して1回の周期をなすように、共通‘タイムクロック’でクロックする。このようにして、シフトレジスタは‘ｎ’ユニット（単位）リアルディレイとして作用する。ここで‘ｎ’はレジスタの長さである（図５４ａ、ディレイレジスタ中の値に従って、マルチプレクサを介して選択される）。代わりに、有限状態マシン（ＦＳＭ）と、1以上のスターティングカウント（starting　count）に対する記憶装置を有するカウンタとを、各ロジック基本要素出力端子と直列に構成する（図５４ｂ）。ＦＳＭは、ロジック基本要素出力状態の変換を検出する。各状態の変換において、発生した特定種類の状態変換（上昇（rising）又は下降（falling））にとって適切なスターティングカウントを用いて、カウンタはＦＳＭによってロードされる。すべてのカウンタは、各時間単位毎に1回の周期を成すように、共通‘タイムクロック’を周期化する。カウントがゼロになった場合、ＦＳＭは、出力状態変換を、ディレイされた出力へと送り、その接続された入力端子へと伝播する（図５４ｃ参照）。

【００９４】３.１.６　リアライザシミュレータから他のシミュレータへの状態の伝送

リアライザロジックシミュレータシステムは極めて高速であるという利点を有し、このため、ソフトウェア又は他のイベントドライブによるシミュレータよりも多くの種々のテストサイクルで処理することができる。このシステムは、ディレイ及び他の時間に関連する項目を表示できず、且つ、設計部中のすべてのノードを監視できないという不利な点も有している。慣用のイベントドライブによるソフトウェアシミュレータはかなり低速であるが、項目の表現及び、刺激及び監視のためのすべての回路網ノードへのアクセスができるという利点を有する。しかし、慣用のイベントドライブによるソフトウェアシミュレータは、非常に低速であるため、シミュレートされた設計部を、初期状態から何１００万又は何１０億周期も離れた誤った状態に送るということは実際には生じない。誤った状態は、実際に、起こり得ないということがわかる。初期状態、すなわち、内部フリップフロップ及びロジックゲート出力の値を読出すことができる(Xilinx ＬＣＡのような）ロジックチップを用いてリアライザシステムを構成する場合、実現されるシミュレーションはストップされ、設計部全体の状態が読出される。リアライザロジックシミュレータと他のシミュレータとを結合させることによって、シミュレートされた設計部の状態（すなわち、設計部中のすべての内部記憶装置の値）が、一方から他方へと伝達される。この際、以下の方法に従っている：

１）同一の設計部を、両方のシミュレータにロードする。

２）リアライザロジックシミュレータ中の設計部は、数サイクルの間、以下のようにシミュレートされる。すなわち、詳細に監視されるべきエラー又は他の条件の発生前の状態へ短時間で設計部を変換する。

３）この時、リアライザ刺激クロックはストップされ、設計部の全状態がロジックチップから読出される。

４）この時、他のシミュレータで表現されている設計部を初期化し、リアライザに基づくシミュレータから読出される状態に適合させる。

５）シミュレーションを、他のシミュレータに進める。

このようにして、リアライザロジックシミュレータの究極的な速度を用いて、長すぎるために、他の方法で取り除くことのできないエラーを除去し、他のシミュレータの詳細及び可視性を用いてエラーの原因を分析することができる。

【００９５】３.２　リアライザフォールトシミュレーションシステム

フォールトシミュレーションとは、テストベクトルを開発、修正するのに用いられるロジックシミュレーションの変形、すなわち、組立て後、設計部、一般的には集積回路の正確性をテストするのに用いられる刺激の組である。ユーザ設計によるフォールティバージョン（faulty version）をテストベクトル刺激を用いてシミュレートするとともに、グッド（good）バージョンと比較し、テストベクトル刺激がグッドバージョンの応答とは別の応答を発生させるかどうかを調べる。別の応答を発生させるならば、テストベクトル刺激が故障（フォールト）を検出していることを示している。故障の多いセットに対しては、このことが繰返される。このことは、できる限り多くの故障を検出する一組のテストベクトルを開発することを目的としている。一般的に、２個の故障を入力設計部の各回路網においてシミュレートする。すなわち、回路網が“スタックアットロー（stack-at-low）”と称する、常にロー状態である場合と、“スタックアットハイ（stack-at-high）”と称する、常にハイ状態である場合とがある。一般的に、入力設計部が何千個もの回路網及びテストベクトルを有し、且つ、フォールトシミュレーションが各新しいテストベクトルのバージョン毎に繰り返されるため、このことは、極めて時間のかかるタスクである。フォールトシミュレーションを構成する新しい手段は、リアライザシステムに基づいている。

リアライザロジックシミュレータの方法を、フォールトシミュレーションの修正に関して用いる。シリアルフォールトシミュレーション技術("Quick Sim Family Reference Manual", Mentor Graphics Corp., Beaverton, Oregon, 1989)を用いる：各故障に関しては：
1）実現された設計部を修正し、故障を伝える。
2）刺激を与え、設計部を作動させ、応答を良好設計の応答と比較し、相異をフラグで合図する。
3）故障を取除き、この故障による相異が存在しているかどうかを記録する。現行のフォールトシミュレーションシステムが、故障設計部の刺激に対する応答を予測するシーケンシャルなアルゴリズムを実行するのに対して、リアライザフォールトシミュレーションでは、実際の故障設計部を実現させ、設計部の刺激に対する応答を決定するという点において、両者は相違する。主な利点は、実現される設計部が、シーケンシャルなアルゴリズムが応答できるよりも速い種々の速度で応答を発することである。
リアライザロジック及び相互接続チップで構成したように、故障は、直接設計部に伝えられる。故障を入力設計回路網に伝えるために：入力設計部の回路網が、ロジックチップ中に対応回路網を有している場合：フォールト構成を用いて、回路網に接続された各ロジックチップを再構成する。これは、回路網に接続された入力端子を、故障に従って、一定のハイ又はローに接続している点を除き、オリジナルの構成と同一である。入力設計部中の回路網が、ロジックチップ中に対応する回路網を有していない場合：対応する回路網はロジックチップのロジック機能に包摂されており、ロジックチップをフォールト構成を用いて再構成する。これは、回路網に包摂されるロジック機能を、回路網が故障に応じて、常にハイ又はローとなるように作動する構成にする点を除き、オリジナルの構成と同一である。故障を取り除くために、オリジナルの構成を用いて、チップを再構成する。リアライザフォールトシミュレータは、以下の相異点を有するものの、リアライザロジックシミュレータと本質的に同様のものである（図55）:リアライザフォールトシミュレータとは、フォールトコンフィギュレータ(configurator)であり、ロジックシミュレータの上位の設計部変換システムの付加的部分を構成している。リアライザフォールトシミュレータは、以下のような各々の故障に対して構成ファイルの相違を出力する：
1）一時的に、故障を設計部データ構造に伝える。
2）どのロジックチップが故障による設計部の変化によって影響されるかを決定する。
3）影響を受けたロジックチップのネットリストファイルを送出する。
4）ＥＲＣＧＡネットリスト変換ツールを用いて、影響を受けたロジックチップの構成ファイルを出力する。
5）フォールト構成ファイルをオリジナルと比較し、構成相違ファイルに相違のみをセーブする。
応答ベクトルメモリを応答回路網に構成する代わりに、設計部コンバータはフォールト応答メモリを構成する。刺激／応答のセクションで説明したように、これらフォールト応答メモリは、応答回路網をメモリ中に記憶された良好な値と比較し、相異が検知される場合にはフリップフロップをセットする。オペレーティングカーネルは、フォールトシミュレーションに対して種々作用する。フォールトシミュレーションを作動させるために（ゼロディレイについて示す。ユニット又はリアル遅延も同様である）：
1）設計部構成ファイルを読出し、これを用いて、すべてのリアライザロジック及び相互接続チップを、構成（configulation）のセクションで説明したように構成する。初期設計部メモリデータを、ファイルから読出し、これを設計部メモリに書込む。構成相違ファイルを読出す。
2）刺激バイナリーファイルを読出す。ベクトルアレイの内容を、ホストインタフェースを介して対応する刺激ベクトルメモリ中に記憶する。タイムアレイ"T"及びサイクルカウント"v"を読出す。良好な回路の応答バイナリーファイルを読出す。対応するフォールト応答ベクトルメモリ中のベクトルアレイの内容を記憶する。
3）この故障の構成の相違を用いて、第1の故障によって影響を受けるロジックチップのフォールト構成ファイルを出力する。またこれらを用いて、この故障に対するロジックチップを構成する。
4）ベクトルメモリモジュール中のすべてのベクトルメモリカウンタ及び相違検出フリップフロップをクリアする。設計部リセット発生器を周期化し、実現される設計部を初期化する。
5）'v'周期のＥＣＬＫ回路網のクロック発生器をイネーブルする。このことによって、刺激ベクトルメモリはこれらの刺激データを送出し、刺激に応じて実現される設計部を作動させることができるとともに、フォールト応答ベクトルメモリは、良好回路に対して応答データを比較する。
6）フォールト応答検出フリップフロップをチェックするとともに、この故障に対して相違が生じたかどうかを記録する。
7）オリジナル構成を、故障したロジックチップに戻す。
8）各残りの故障に対して、ステップ3）～7）を繰り返す。

【００９６】3.3　リアライザロジックシミュレータ評価システム

現在のＥＤＡシステムにおける現行の慣用的なシミュレータの多くは、イベントドライブと称するよく知られたシーケンシャルなアルゴリズム、又はコンパイルされたコードシミュレーションのいずれかに従って作動する

("An Introducion to Digital Simulation",Mentor Graphics Corp.,Beaverton,Oregon,1989)。第1アルゴリズムにおいては、入力設計部における各基本要素を各時間ステップ毎に"評価"する。この場合、基本要素の入力ピンを駆動する回路網は、イベント即ち状態の変化を有している。また第2アルゴリズムにおいては、全ての時間ステップに対して入力設計部の各基本要素を評価する。基本要素の評価とは、基本要素の新しい出力値が新しい入力値に対してどのようなものであるかを決定する動作のことである。このことは、シミュレーションの間何回も生じる。通常、ゲートのような小さな基本要素のみを1オペレーションで評価する。この際、索引テーブル又は他の直接的な技術を使用する。大規模ロジック回路網は、一般的に小さな基本要素及び回路網の組み合わせとしてシミュレートされる。多くの時間のかかる内部評価は、各大規模回路網の評価毎に必要とされる。リアライザシステムの外部にあり、シーケンシャルなシミュレーションアルゴリズムを実行するロジックシミュレータをリアライザロジックシミュレータ評価システムに結合させる。これは、リアライザのハードウェアを用い、アルゴリズムシミュレーション中の1以上の大規模ロジック回路網を評価する。リアライザシステムによって評価されるべき各大規模ロジック回路網を、外部ロジックシミュレータ中に単一基本要素で表現する。この利益の一つはそのスピードにある。その理由は、実現された基本要素がほとんど瞬間的に評価されるからである。リアライザシステムによって評価されるロジック回路網のサイズは、リアライザのロジック容量によってのみ制限され、全入力設計部と同量のロジック容量を包含している。リアライザロジックシミュレータ評価システムは、リアライザハードウェアシステム及びホストコンピュータと相俟って、リアライザ設計部変換システム(既に記載した)と、リアライザロジックシミュレーションイバリュエータ(evaluator)とからなっている(図56)。これを、シーケンシャルなシミュレーションアルゴリズムを作動させる外部ロジックシミュレータに結合させる。リアライザロジックシミュレーション評価システムによって、評価のためのロジック回路網を準備するために:

1)リアライザシステムによって評価されるべきロジック回路網をEDAシステムの入力設計部として組み立てる。

2)特性を各ロジック回路網の入出力回路網に組込み、シミュレータ及びサンプラによってそれぞれ駆動されるように指示する。

3)通常の方法でリアライザ設計部変換システムを用い、入力設計部を変換し、ロジック回路網のこの集合体に構成及び対応テーブルファイルを出力する。

シミュレーションを行うために、以下の方法に従って、外部ロジックシミュレータを作動させ、シミュレータアルゴリズムを実行させると共にリアライザロジックシミュレーション評価装置も作動させる。:

1)外部シミュレータのデータ構造を構成し、リアライザシステムによって評価されるべき各ロジック回路網毎に単一の基本要素を設ける。

2)設計部の対応テーブルファイルを読み出し、基本要素入出力を、リアライザホストインターフェースバスのこれらのアドレスと関連させる。

3)構成のセクションにて述べたように、設計部の構成ファイルを読み出し、これを用いて全てのリアライザロジック及び相互接続チップを構成する。ファイルから初期設計部メモリデータを読み出し、これを設計部メモリに書き込む。設計部リセット発生器を周期化させ、実現されるロジック回路網を初期化する。

4)初期値を用いて全てのシミュレータを初期化する。

5)外部ロジックシミュレータのシミュレーションアルゴリズムを作動させる。シミュレーションアルゴリズムでは、この方法を用いてリアライザに基づく基本要素を評価する:

1)このシミュレーション時間ステップにおける、この基本要素への全ての入力に対する値を、リアライザロジックシミュレーションイバリュエータに伝送すると共に、この値をロードするために、対応するシミュレータに送る。

2)リアライザロジックシミュレーションイバリュエータにこの基本要素の全ての出力サンプラをチェックするよう指示し、いかなる出力に対する変化であってもシミュレーションアルゴリズムに伝送し直す。

6)シミュレーションの前後において、ユーザが試験及び修正を行うために、ホストインターフェースを介して設計部メモリ内容をアクセスするために、外部ロジックシミュレータのユーザインターフェースシステムの機能を与える。

シミュレーションアルゴリズムをソフトウェア中で実行する場合、これをリアライザホストコンピュータで実行するとともに、ホストインターフェースを用い、シミュレータ、サンプラ及び設計部メモリをアクセスする。シミュレーションアルゴリズムをハードウェアで実行する場合、ホストコンピュータへの通信リンクを用い、シミュレータサンプラ及び設計部メモリをアクセスする。ハードウェアシミュレータシステムの変更には、シミュレータハードウェアとリアライザのユーザ指定によるデバイス(USD)モジュールとの間の直接接続を用いる。この方法は以下の相違点を伴うものの、上記と同様である:

1)入力設計部の基本要素の入出力に関するシミュレータ及びサンプラを指示する代わりに、これらを、ハードウェアシミュレータの評価ユニットに対応するUSD基本要素に接続する。

2)ハードウェアシミュレータの評価ユニットを、リア

ライザのＵＳＤＭに電気的に接続する。入力イベントが発生すると、新しい値を直接接続によって実現される基本要素に供給すると共に、ホストを介するのではなく直接接続によって出力応答を捕捉する。このため、かなりの高速評価スピードが得られる。

【００９７】３．４　リアライザプロトタイピングシステム

入力設計部を実現する場合、これを直接設計部のプロトタイプとして実現し、作動させることができる。一般的にリアライザシステムのタイミングディレイは、究極的なハードウェアの実現によるタイミングディレイと一致しておらず、このためプロトタイプはフル設計スピードで作動することはできないが、リアライザベースのプロトタイプによって、ほとんど実時間で設計部は実際に動作することができる。実現される設計部を、リアライザクロック発生器、ホストを介して制御されるシミュレータ、実際にユーザが指定するハードウェアデバイス、実現される仮想計器（以下で説明する）によってシミュレートし、及び／又は、内部ロジック及び／又は設計部メモリ内容によって自己シミュレートする。設計部のオペレーションを、ホスト、実際にユーザが指定するハードウェアデバイス、実現される仮想計器を介して、及び／又は、設計部メモリ内容を調べることによって、コントロールされるサンプラを用いてモニタするとともに分析する。設計者は直接、“ベンチトップ（benchtop）”環境と同様に、実時間で設計部と対話する。リアライザプロトタイピングシステムは、リアライザハードウェアシステム及びホストコンピュータとともに、設計部変換システムと、プロトタイピングシステムとを具えている（図５７）。プロトタイピングオペレータは、作動される設計部のリアライザシステムを構成し、リアライザ設計部の対話型刺激及び応答をサポートする。このオペレータはホストコンピュータにおいて実行し、直接又はホストコンピュータにおいてランする制御プログラムを介して、ユーザのコマンドに対して応答する。実現される設計部を作動させるために：

１）構成のセクションにて述べたように、設計部の構成ファイルを読み出し、これを用いて、全てのリアライザロジック及び相互接続チップを構成する。ユーザが供給するファイルから初期設計部メモリデータを読み出し、これを設計部メモリに書き込む。対応するテーブルファイルを読み出すと共に、設計部回路網ネーム間の対応と、スティミュレータ及びサンプラ及びこれらのホストインターフェースバスアドレスとを確立する。

２）設計部リセット発生器を周期化し、実現される設計部を初期化する。

３）連続的に以下の動作を必要に応じて行う：−ユーザコマンドを処理し、クロック及びリセット発生器を制御する。−ユーザコマンドを処理し、スティミュレータ出力値を変化させる。この際、対応テーブルを用い、ユーザが与える回路網ネームを、対応するスティミュレータと関連させる。−ユーザコマンドを処理し、サンプラのデータ入力値を表示する。この際、対応テーブルを用いユーザが与える回路網ネームを、対応するサンプラと関連させる。−ユーザコマンドを処理し、設計部メモリモジュール中のロケーションを読み出すと共に書き込む。設計部が動作していないことを確認する。この際、設計部メモリをアクセスする前にクロック発生器が停止し、不適切な設計部メモリの動作を回避することをチェックする。設計部が停止されていないかどうかを、ユーザに報告する。リアライザプロトタイピングシステムを使用するために：

１）入力設計部をホストＥＤＡシステム中に作成する。

２）シミュレータに接続されるべき設計部回路網と、サンプラと、クロック又はリセット発生器とをマークする。

３）設計部基本要素、回路網及び接続を設け、用いるべき任意の仮想計器に対する回路網を設計する（以下参照）。

４）リアライザ設計部変換システムを用いて入力設計部を変換し、設計部の構成ファイルを出力する。

５）リアライザプロトタイピングオペレータを用いて、設計部を作動させる。図５７にて示す特定の例においては、デジタルコンピュータ設計部を、リアライザプロトタイピングシステムを用いて実現する。ユーザは、ホストＥＤＡシステムを用いて、入力設計ファイル中のコンピュータロジック及びメモリの設計部を表現し、ユーザは、リアライザ設計部変換システムを用いて、構成ファイルへと変換する。実際の具体例においては、実際のフロントパネル制御スイッチ及びインジケータに接続されているフロントパネル制御入力及びディスプレイ出力は、入力設計部において指定され、プロトタイプオペレータを介してのユーザ制御の下、スティミュレータ及びサンプラに接続される。コンピュータのクロック入力信号がリアライザクロック発生器によって出力されるように指定する。

プロトタイプコンピュータを作動させるために、ユーザはリアライザプロトタイプオペレータをランさせ、コンピュータ設計に応じてリアライザシステムを構成する。実現されるコンピュータ設計部で実行が可能となるように、コンピュータプログラムコードをロードすると共に、その初期データを、プロトタイプオペレータを介し、動作の開始時に設計部メモリへロードする。ユーザがクロック発生器をイネーブルさせると、コンピュータ設計部は、リアライザハードウェアの構成されたロジック及び相互接続チップにおいて実際に動作し、設計部メモリから読み出されるプログラムインストラクションコードを実行するとともに、設計部メモリ中のデータを読み出し且つ書き込む。ユーザは、フロントパネル制御入力端子を作動させ、プロトタイプオペレータの対応ステ

ィミュレータ及びサンプラへのアクセスを介して、動作中ディスプレイ出力を読み出す。結果はプログラムの終了時に、プロトタイプオペレータを介してメモリ中からユーザによって読み出される。ユーザはこの結果を解析し、設計部が正確であるかどうか、すなわちユーザの意図に従って正しく動作しているかどうかを判断する。入力設計部中の設計エラーのために設計部が正しく動作していない場合、ユーザは、ホストＥＤＡシステム用いてエラーを修正し、プロトタイピングプロセスを繰り返す。

【００９８】３.４.１　実現される仮想計器

刺激及び／又は分析計器が、プロトタイプデバッギングプロセスにおいて必要とされる場合、ロジックアナライザのような慣用の計器を、ユーザが与えるデバイスモジュールを介して、実現される設計部に直接接続される。実際の計器を接続するために、計器に接続されるべき設計回路網に接続され、入力設計部中の計器ＵＳＤを表示している基本要素を設けるとともに、ＥＳＤ接続を規定しているＵＳＤ使用ファイルを作成する。このとき、計器を直接ＵＳＤＭに接続し、上記のように実現される設計部を変換させ、作動させる。さらに、“仮想計器”を、入力設計ファイル中の設計部に設け、且つ、この設計部を用いて実現される基本要素及び回路網を設けている。例えばロジックアナライザーを、一組のロジック信号をモニタするよく知られた計器とし、これらが一定のトリガ条件を満足する場合、一組の分析された信号を連続的にサンプリング化するとともに、これらの値をメモリ中に記録する。これはその後分析のために読み出される。図５９は、仮想ロジックアナライザの構成を示し、このアナライザは、応答ベクトルメモリと、ロジック基本要素を有する条件検出器と、１個以上のスティミュレータ及びサンプラと、他のロジック基本要素とを具えている。設計部を用いて仮想ロジックアナライザを実現及び使用するための：

１）設計部に加えて、図で示したように、相互接続された入力設計ファイル中のこれら成分に対する基本要素を設ける。特に、応答ベクトルメモリ入力を、分析されるべき設計部回路網に接続し、条件検出器入力端子を、トリガ条件でモニタされるべき設計部回路網に接続し、これによって、検出されるべき条件に従って条件検出器のロジックを指定する。

２）入力設計ファイルを、通常の手続きに従って構成ファイルに変換する。

３）リアライザプロトタイピングシステム中の設計部を構成する。

４）シミュレータを介して“リセット”信号を周期化させ、実現される設計部が動作を開始するのに必要とされる刺激を与える。

５）“トリガされた”サンプラをモニタする。サンプラが“トリガされた”信号が真であることを示している場合、ロジックアナライザは分析された信号データを捕捉する。

６）このデータを、ロジックアナライザの応答ベクトルメモリからホストインターフェースを介して読み出す。これを、一般的なコンピュータデバッガプログラム又はこれと同様のものを用いて、表示するとともに分析する。

これは、いかにして仮想刺激又は分析計器を、リアライザシステム中の設計部を用いて実現するかを示す一例である。ロジックアナライザの概念のような、計器自体の概念が新規でないことに注意する。リアライザシステム中の入力設計部を用いて計器を実現することが、新規性の一要素となっている。

【００９９】３.５　リアライザ実行システム

リアライザ実行システムを用い、入力設計ファイル中で指定され、未だ構成されていない又は永久的ハードウェアにおいて、決して構成することを試みることはないハードウェア機能を実行する。このことを行うことによって、幾つかの利点が得られる：永久的ハードウェアを構成する間、ソフトウェア開発又はその他の目的のために、実現される設計部を使用する。このことによって、例えば、ソフトウェア開発に作成中に行うことが可能となり、これをデバッグし、永久的ハードウェアを使用しない場合にソフトウェアを使用できるように準備する。リアライザ実行システムはユニバーサルハードウェアデバイスとしての役割を果たし、必要とされる種々の互換を行うために用いられる。特別な機能が要求される場合（リアライザ設計変換システムによって実現される場合）、ハードウェアシステムの構成ファイル及びその他のファイルは、ホストコンピュータによって記憶装置から呼び出され、リアライザシステムをこの設計に応じて構成し、機能を実行する。例えば、電気的な設計環境では、リアライザ実行システムを用いて、必要とされるロジックシミュレーションハードウェアアクセレータ、ルーティングハードウェアアクセレータ、又はハードウェアグラフィックスプロセッサの役割を果たす。デジタル信号を処理する環境において、リアライザ実行システムを用いて、必要とされる実時間スペクトラムアナライザ又は特別な効果を有するシンセサイザの役割を果たす。リアライザ実行システムは以下の点を除き、リアライザプロトタイピングシステムと同様のものである：

１）分析のための計器を用いず、入力設計を正しいものとみなす。スティミュレータ、サンプラ、及び設計メモリアクセスのみを用いて実行する役割を制御し、データを入出力する。

２）特定の実行される機能を指示するコントローラを作成することができ、又、これを用いてリアライザプロトタイピングオペレータを制御し、実行する機能を、機能の使用に適した入力端子／出力端子及び制御インターフェースに与える。

【0100】3.6 リアライザ生産システム
リアライザ設計変換システムの変形例を用いて、自動的に入力設計部の永久的で再構成不可能な実現例を作成する。この永久的な実現例では、実現される設計部に構成されるのと同じ種類と数のリアライザロジックチップを用いる。リアライザ生産システムでは、そのERCGAネットリスト変換ツールを用い、機能においてERCGAロジックチップと等価な、永久的で再構成することのできないロジックデバイスを構成するとともに、自動的プリント回路ボード(PCB)配置及びルーティングツールを駆動する("Getting Started with Board Station", "Layout User's Manual",Mentor Graphics Corp.,Beaverton, Oregon.1989)。この際、ロジックチップ相互接続に関する仕様を用い、これら再構成することのできないロジックデバイスを永久的に相互接続するPCBを製造する。好適例では、LCAをERCGAロジックチップとして使用している。LCAを製造することによって、機能的にLCAと等価な再構成することのできないロジックチップを、構成PROMメモリと結合しているLCAチップの形態で提供する("The Programmable Gate Array Data Book",Xilinx, Inc.; San Jose,1989)。LCAネットリスト変換ツールによって、PROMをプログラムするのに用いられるバイナリーファイルを作成する。また、LCAはロジックを具え、これを用いてLCAが電力を供給する際にLCA自体を構成することができる。この際、PROMがあればこれを用いる。リアライザ生産システムは、前述したのと同一の設計部リーダと、基本要素コンバータと、リアライザ設計部変換システム(RDCS)、相互接続及びネットリスティングシステム及びRDCS中に用いられるものの変形であるERCGAネットリスト変換ツールに使用されるパーティショナと、自動PCB配置と、ルーティングツールとを具えている(図60)。リアライザ生産システムは、リアライザハードウェアシステム又はホストコンピュータを具えていない。これは入力設計部ファイル及びPCB仕様ファイルを読み出す。以下の方法に従って作動する:
1)設計部リーダを用い、入力設計部ファイルを読み出すとともに、設計部データ構造を作成する。
2)基本要素コンバータを用いて、設計部データ構造をロジックチップ基本要素へと変換する。
3)パーティショナを用い、基本要素を特定のロジックチップに割り当てる。
4)相互接続及びネットリスティングシステムを用い、ロジックチップのためのネットリストファイルを作成する。相互接続チップのためのネットリストファイルを供給する代わりに、カット回路網及びこれらのロジックチップI/Oピン接続のリストを、自動PCB配置及びルーティングツールに受け入れられる形態で単一の相互接続ファイルを送出する。
5)ERCGAネットリスト変換ツールを用い、再構成することのできない等価ロジックデバイスの構成に適した形態で、各ロジックチップ毎にバイナリー構成ファイルを供給する。
6)自動PCB配置及びルーティングツールを用いて、相互接続ファイル及びPCB仕様ファイル(このファイルは、PCBの寸法、コネクタ必要条件などのようなロジック設計とは直接関係していない物理的な情報を具えている)に読み込み、PCB製造データファイルを出力する。
リアライザ生産システムのユーザは、このようにしてPCB製造データファイルを用いて、PCBを製造し、バイナリー構成ファイルを用いて、再構成することのできないロジックデバイスを構成すると共に、デバイス及びPCBを組立て、入力設計部の最終的な実現例を提供する。リアライザ生産システムにおいて、機能的に永久的ハードウェアの実現例のERCGAと等価な再構成することのできないゲートアレイチップを使用することは新規なことではなく、一般的に行われていることである。むしろ、このシステムが任意の大きさのデジタルシステムを作り出すことができるということ(これは、1個のICチップの容量に限られることではない)、このシステムが入力設計ファイル中で包括的な基本要素ロジックの形態で表現されること(特定のコンピュータメーカのロジックライブラリには限定されない)と、更には、自動的に永久的ハードウェアの実現例を提供するということ、これらが新規性の一態様といえる。
【0101】3.7 リアライザ計算システム
リアライザハードウェアシステムを、パスカルのような高級コンピュータ言語で書かれている入力プログラムで特定される動きに従って構成することができる。又これを用いて、コンピュータの実行する一般目的のための記憶された記憶されたプログラムに従って計算機能を実行することができる。このことは、高レベル設計合成コンパイラを用いることによって達成され、コンピュータプログラムを、入力設計ファイル中に表現されているデジタルロジックの形態に変換し、その後リアライザハードウェアにおいて、この設計部を実現すると共に作動させる。この方法は根本的に新しい計算手段である。計算の見地から見れば、リアライザハードウェアを高度並列処理データプロセッサとし、そのデータプロセッシング素子を、リアライザロジックチップ、相互接続チップ及び特定目的のエレメント中のロジック機能及び記憶デバイスである。このデータプロセッサは、シーケンシャルな計測を行うことに関する記憶されたプログラム計算方法に従って演算を行うわけではない。このデータプロセッサは、リアライザハードウェアに構成され、入力プログラムで指示される動きに従って作動するデータパスと、機能的ユニットと、有限状態マシン制御構造とに従って作動する。この利点は、計算スピードがシーケンシャル

に記憶されたプログラムによる計算で可能な計算スピードよりも速いことである。説明するリアライザ計算システムは、リアライザハードウェアシステム及びホストコンピュータとともに、リアライザ計算コンパイラと、リアライザ設計部変換システムと、リアライザ計算オペレータとを具えている（図６１）。このホストコンピュータが、ただリアライザ計算オペレータをランさせる手段としてのみ用いられ、入力プログラムで指示される計算機能を実行することに関しては用いられないということに注意する。リアライザ計算オペレータをランさせる他の手段を用いることができること勿論である。

【０１０２】３.７.１　リアライザ計算コンパイラ

リアライザ計算コンパイラは、テキストエディタを用いて高級言語で書かれた入力プログラムファイルを、入力設計部ファイルに変換する。これは、設計部合成コンパイラと、ロジック合成コンパイラと、機能的ユニットコンパイラとを具えている。設計部合成コンパイラは、ツールであり、その幾つかの例は最近開発されたものである（“Tutorial on High-Level Synthesis”,McMarland,Parke and Camposano, Proceeding of the 25th Design Automatic Conference, ACM and IEEE,1988)。このコンパイラは、機能的ユニット、データ入出力から成る有限状態マシンコントローラ及びデータパスのシステムとバス相互接続とに関する記述を構成し、標準的な手続コンピュータ言語で特定される動きに従って作動する。実際の設計部合成コンパイラの一例としては、“フラメル（flamel）”がある。その方法については、“Flame l:A High-Level Hardware Compiler”,Howard Trickey, IEEE Transaction on Computer-Aided Design, Vol. CAD-6, No.2, 1987)で詳細に説明されている。文献からの引用を示す：“フラメルへの入力は、パスカルプログラムである。”“ユーザは、パスカルプログラムに、入力プログラムを一般的に実行する場合の実行頻度を与える。その他のユーザ入力は、どの程度のハードウェアが許容されるかを示すナンバである。出力は、パスカル言語と同一の役割を果たすハードウェアの設計である。”“フラメルによって作り出されるモデルとは、データパス及びコントローラから成る同期デジタルマシンである。データパスは、バスによって相互接続される機能的ユニット（ＡＬＵ、加算器、レジスタ、Ｉ／Ｏパッド等）から成っている。コントローラは有限状態マシンである。”“一般的なパスカルプログラムを用いて、ハードウェアに要求される動きを規定する。フラメルは、プログラム中の並列処理を見出し、ユーザ指定によるコスト制約と合致した高速実行の実現を可能としている。”“フラメルの実現例は完成されている。出力は、データパス及びコントローラに関する記述である。一連のテストにおいて、フラメルは、クロックサイクルを同じとすると、同じプログラムを実行するのにMC68000(マイクロコンピュータ）の２２～２００倍の速さでランするプログラムを具体化する。”用いられるリアライザハードウェアシステムの容量に従って、ユーザ又はリアライザ計算システムは、“ユーザ指定のコスト制約がある”入力を、この設計部合成コンパイラに供給する。設計部合成コンパイラの出力は、データパス及びコントローラの記述を具える中間表現ファイルである。機能的なユニットライブラリとは、一組の予め定義された機能的なユニットの表現である。各タイプの機能的なユニットに対する表現は、設計部合成コンパイラによって与えられる。これらの表現は、ロジック及びユーザ指定のデバイス(ＵＳＤ）基本要素と、これらの回路網相互接続とを指定する。これらの表現は、リアライザ入力設計部基本要素の要件と合致している。ＵＳＤ基本要素は、付加的に用いられ、ロジックチップ及び設計部メモリを用いて実現されるものよりも、より高い性能又は、より大きな容量の基本要素を提供することができる。例えば、高速ＶＬＳＩ浮動小数点型乗算器をＵＳＤとして取り付ける場合、機能的なユニットバイナリーは、このＵＳＤ基本要素を特定する浮動少数点型乗算器の機能的ユニットに関する記述を具えている。ロジック合成コンパイラは、データパス及び有限状態マシンコントローラに関する記述を、入力計算ファイル中のロジック基本要素及び相互接続回路網に関する表現に変換する。このロジック合成コンパイラは、有限状態マシン合成ツールを具え、これは、Mentor Graphics Corp.,VLSI Technology Inc.,Synopsis Inc.等（“Logic Synthesis speeds ASIC Design”, A. J. de Geus, IEEE Spectrum,August 1989 ）から商業的に入手可能であり、又は、文献記載の方法に従って開発される（“The Implementation of a State Machine Compiler”, C. Kingsley,Proceedings of the 24th Design Automation Conference, ACM and IEEE,1987; "A State Machine Synthesizer",D. Brown, Proceedings of the 18th Design Automation Conference, ACM and IEEE, 1981; "An Overview of Logic Synthesis Systems", L.Trevillyan, Proceedings of the 24th Design AutomationConference, ACM and IEEE, 1987)。このコンパイラは、以下の方法に従って作動する：

１）データパス及びコレクタに関する記述を含む中間表現ファイルを、データ構造中に読み込む。

２）機能的ユニットライブラリの記述に従って、各データパスの機能的なユニットの記述を、ロジック及びＵＳＤ基本要素及び回路網に変換する。

３）データパスへの各データ入力及び、データパスからの各データ出力に対する設計部メモリ基本要素を提供する。

４）有限状態マシン合成ツールを用いて、有限状態マシンコントローラの記述を、ロジック基本要素及びこれらの回路網相互接続に変換する。

５）有限状態マシンコントローラへの‘スタート’入力と、有限状態マシンコトローラからの‘ビジー’及び

'ダーン(done)' 出力とに対するスティミュレータ及びサンプラの基本要素を提供する。
6)クロック回路網が、リアライザクロック発生器によって駆動されるように指示する。
7)基本要素及び回路網を、入力設計ファイルに送出する。
【0103】3.7.2 リアライザ計算オペレータ
リアライザ計算オペレータは、リアライザシステムを構成し、本来的に入力プログラムが指示し、実現される計算機能の実行を可能にする。リアライザ計算オペレータは、設計変換によって作成される構成ファイル及び対応テーブルファイルに読み込み、ユーザ指定の計算機能への入力データに関するファイルを読み出すとともに、計算機能からの出力データに関するファイルに書き込む。実現される計算機能を作動させるために:
1)設計部の構成ファイルを読み出し、これを用いて、構成のセクションにて述べたように、すべてのリアライザロジック及び相互接続チップを構成する。
2)入力データファイルを読み出し、そのデータを入力データ設計部メモリに書き込む。出力データ設計部メモリをクリアする。
3)対応テーブルファイルを読み出し、コントロール入力と出力との間、スティミュレータとサンプラとの間、及びホストインターフェースバスアドレス間の対応を決定する。
4)クロック発生器をイネーブルし、スティミュレータを介して'スタート'制御入力を主張し、動作を開始させる。
5)'ダーン'制御出力をモニタし、これが真となる時に、データを出力設計部メモリから読み出し、これを出力データファイルに書き込む。リアライザ計算システムを用いるために:
1)テキストエディタ又は他の手段を用いて、入力プログラム及び入力データファイルを準備成する。
2)リアライザ計算コンパイラを用い、入力設計ファイルを発生させる。
3)他で既に述べたように、通常の方法で作動するリアライザ設計変換システムを用い、構成及び対応テーブルファイルを発生させる。
4)リアライザ計算オペレータを用い、実際に計算機能を実行する。
5)実現された計算機能によって計算されるデータを、出力データファイルから読み出す。
【0104】4 好適例
この明細書を介して説明される好適例は、以下の特徴を有している:
【0105】4.1 ハードウェア
部分的クロスバー相互接続を、3レベルで階層的に、全ハードウェアシステムに用いる。図62-図64は、階層的に相互接続されたロジックボード、ボックス及びラックの一般的アーキテクチャを示している。図65a-bは、ボード、ボックス及びラックに関する物理的構造を示している。
ロジックボード(図62):各ロジックボードは、32個のXレベルクロスバーチップによって相互接続された14個のLチップから成っている。各Lチップは、Xレベルクロスバーに接続されたチップ毎に128個のI/Oピンを有しており、32個のXチップの各々に4個の接続が成されている。14個の付加的なI/Oピンを用いる;その内11個をRバスに接続し、1個を2個のクロック信号の各々に接続し、1個を設計部リセット信号に接続する。Xilinx XC3090 LCAをロジックチップとして用いる。各Xチップは、ロジックチップに接続された56個のI/Oピンを具えており、14個のLチップの各々に4個の接続がなされている。各Xチップは、2個のYチップの各々と、8個の付加的I/Oピン接続を具えている。Xilinx XC2018 LCAをXチップとして用いる。各ロジックボードは、X-Yバスに対し512個の背面I/Oピンを有している。これは、Rバス及び構成バスへの接続も有している。
ボックス(図63):各ボックスは、64個のYレベルクロスバーチップによって相互接続されている1~8個のボードから成っている。各Yチップは、ロジックボックスボードに接続された6個のI/Oピンを有しており、各ボードのXチップに8個の接続が成されている。これは、1個のZチップとの8個の付加的なI/O接続を有している。Xilinx XC2018LCAをYチップとして用いる。64個のYチップを、8個のYチップボードに取り付ける。この各々は、X-Yバスに対して512個の背面I/Oピンを有している。8個のYチップボードと8個のロジックボードとを、ボックスのX-Yバス背面のワイヤで相互接続する。各Yチップボードは、Y-Zバスに対して、ケーブルコネクタに64個のI/Oピンを有している。各ボックスは、このようなコネクタを8個有している。これらの接続を、各ボックスからの単一な512個のワイヤY-Zバスケーブルに集める。各Yチップボードは、構成バスに対する接続も有している。図65aは、Y-Zバスケーブルを具え、ホストインターフェース、8個のロジックボード及び8個のYチップボードを有しているX-Yバス背面の物理的な構成を示している。
ラック(図64):各ラックは、1~8個のボックスを具え、64個のZレベルクロスバーチップによって相互接続されている。各Zチップは、ボックスに接続された64個のI/Oピンを具え、各ボックスのYチップに8個の接続がなされている。Xilinx XC2018 LCAをZチップとして用いる。ラックのボックスを、ロジックボードに配置された各ボックスからのX-Zバスケーブルへの接続を用いて、付加的なボックスによって相互接続する。図65bにおいて、Zレベルボックスの物理的な

構成を示す。６４個のＺチップを、８個のＺチップボードに取り付ける。この各々は、Ｙ－Ｚバスに対して５１２個のＩ／Ｏピンを有している。８個のＺチップボードと、８個のＹ－Ｚバスゲーブルコネクタとを、Ｙ－Ｚバス背面のトレースによって相互接続する。メモリのセクションにて述べたように、１６個のＲＡＭチップと１０個のＬＣＡとを各々具えているメモリモジュールを、必要とされる場所であるロジックチップの場所に取り付ける。メモリモジュールは、刺激及び応答のセクションで規定されているように、設計部メモリ、ベクトルメモリ、スティミュレータ及びサンプラに用いられている。ユーザ指定によるハードウェアデバイスモジュールをロジックチップＬＣＡの場所に取り付ける。ある１個のボックスは、ホストコンピュータのＩ／Ｏバスインタフェースカードとケーブル接続しているホストインターフェースボードを具えている。このボックスは、Ｒバスと称するホストインターフェースバスを制御する。すべての制御及びデータ伝送機能のためにこのバスをすべてのロジックチップロケーション及び各ロジックボード、すなわちＹチップボード及びＺチップボードにおける構成制御ロジックブロックに接続する。Ｒバスは、そのセクションで述べたように、８ビットデータバスと、クロックと、２個のコントロールラインとを具えている。ホストインターフェースボードは、構成バスコントローラと、２個のクロック発生器と、リセットコントローラとを具えている。１６ビットデータバスを有する構成バスは、全ての構成機能に対して、ホストインターフェースを用いて、すべてのロジック及びクロスバーチップを接続する。各ボードの１４個のＬチップを一つの構成グループとし、その３２個のＸチップを２つのグループに分割する。８個のＺチップボードの各々と同様に、各ボックスの８個のＹチップボードを各々１グループとする。

【０１０６】４．２ソフトウェア

設計部変換システムは以下のモジュールから成っており、その各々は、それぞれに関するセクションで記載されている：Quick Sim ロジック基本要素を有する Mentor Graphics設計ファイルを読み出す設計部リーダ。Quick Sim 基本要素をXilinx LCA基本要素に変換する基本要素コンバータ。トライステート及びワイヤードネットドライバは、トライステートのセクションで述べたように、クロスバー加算構成に従って変換される。そのセクションにて述べたように、クラスタ構成技術に基づいているパーティショナ。３つのレベルの部分的クロスバーを相互接続するとともに、システム中の各ロジック及びクロスバーチップに関するＸＮＦフォーマットのネットリストファイルを送出する相互接続及びネットリスティングシステム。XNF2LCA, APR及びMakebitsから成っているXilinx ＬＣＡネットリスト変換ツール。構成ファイルコレクタ。

応　用

Mentor Graphics のログファイルに基づき、且つＲＳＩＭバッチインターフェースツールを用いているリアライザロジックシミュレーションシステム。Mentor Graphics のログファイルに基づき、且つＲＳＩＭバッチシミュレーションインタフェースツールを用いているリアライザフォールトシミュレーションシステム。Mentor Graphics Quick Sim のロジックシミュレータとして作用するリアライザロジックシミュレータ評価システム。ロジックアナライザを具え、実現される仮想計器を有しているリアライザプロトタイピングシステム。

リアライザ実行システム

Mentor Graphics Board Station の自動ＰＣＢ配置及びルーティングツールを用いてのリアライザ生産システム。パスカル言語とフラメル設計部合成コンパイラと、Mentor Graphics Design,Knowledge and Logic Consultant FSM 及びロジック合成ツールとを用いたリアライザ計算システム。好適例を引用し、本発明の原理を説明したが、このような原理とは離れて、装置及び細部を種々変更できること明らかである。例えば、Mentor Graphics の電気設計オートメーションの変形例を用い、本発明が有効に動作することを説明したが、他の設計部オートメーションツールを用いて同様に本発明を実施できることがわかる。本発明は、ここに開示されている実施例に限定されるものではなく、要旨を変更しない範囲内で種々の変形や変更が可能である。

【図面の簡単な説明】

【図１】図１は、リアライザハードウエアシステムを示す略ブロック図である。

【図２】図２は、直接相互接続システムを示す略ブロック図である。

【図３】図３は、チャンネル・ルーティング相互接続システムを示す略ブロック図である。

【図４】図４は、クロスバー相互接続システムを示す略ブロック図である。

【図５】図５は、クロスバー回路網相互接続システムを示す略ブロック図である。

【図６】図６は、部分的なクロスバー相互接続システムの簡単な一例を示す略ブロック図である。

【図７】図７は、部分的なクロスバー相互接続システムを示す略ブロック図である。

【図８】図８は、クロスバーチップ幅の相違を説明するための図である。

【図９】図９は、トライステート回路網を示す略ブロック図である。

【図１０】図１０は、図９のトライステート回路網と等価な、積の和を示す略ブロック図である。

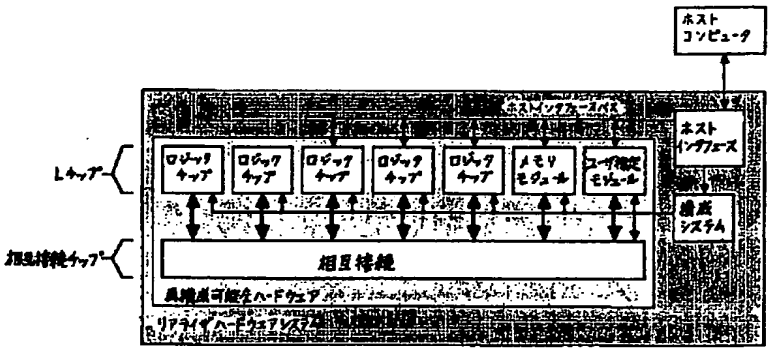
【図１１】図１１は、“フローティングロー”及び“フローティングハイ”の積の和の回路網を示す略ブロック図である。

【図１２】図１２は、相互接続を最小とするように構成

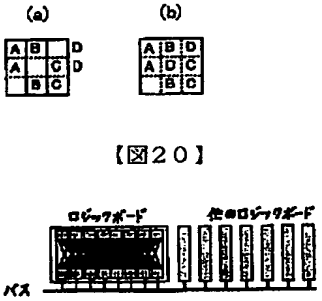
されたドライバ及びレシーバを示している略ブロック図である。
【図１３】図１３は、ロジック加算構成を示している略ブロック図である。
【図１４】図１４は、クロスバー加算構成を示している略ブロック図である。
【図１５】図１５は、双方向性のクロスバー加算構成を示している略ブロック図である。
【図１６】図１６は、双方向性のクロスバートライステート構成を示している略ブロック図である。
【図１７】図１７は、部分的なクロスバーからのオフボード接続を示している略ブロック図である。
【図１８】図１８は、Ｙレベルクロスバー相互接続を示している略ブロック図である。
【図１９】図１９は、双方向性バスのシステムレベル相互接続を示す略ブロック図である。
【図２０】図２０は、共通バス相互接続に基づく、８個のボードを示す略ブロック図である。
【図２１】図２１は、２つのバスレベルの階層を示す略ブロック図である。
【図２２】図２２は、最大バス相互接続階層を示している略ブロック図である。
【図２３】図２３は、汎用メモリモジュール構造を示す略ブロック図である。
【図２４】図２４は、メモリアドレスロジックチップを示す略ブロック図である。
【図２５】図２５は、共通Ｉ／Ｏを用いてのメモリデータロジックチップを示す略ブロック図である。
【図２６】図２６は、分離Ｉ／Ｏを用いてのメモリデータロジックチップを示す略ブロック図である。
【図２７】図２７は、一個のデータビットに関する多重ＲＡＭを示す略ブロック図である。
【図２８】図２８は、メモリモジュールの好適例を示す略ブロック図である。
【図２９】図２９は、刺激ベクトルメモリを示す略ブロック図である。
【図３０】図３０は、応答ベクトルメモリを示す略ブロック図である。
【図３１】図３１は、刺激及び応答に対するベクトルメモリを示す略ブロック図である。
【図３２】図３２は、ベクトルメモリアドレスチップの好適例を示す略ブロック図である。
【図３３】図３３は、ベクトルメモリデータチップの好適例を示す略ブロック図である。
【図３４】図３４は、ランダムアクセススティミュレータを示す略ブロック図である。
【図３５】図３５は、エッジ感応タイプのスティミュレータを示す略ブロック図である。
【図３６】図３６は、サンプラの略ブロック図である。
【図３７】図３７は、変更検出サンプラを示す略ブロック図である。
【図３８】図３８は、ユーザ指定のデバイスモジュールアーキテクチャを示す略ブロック図である。
【図３９】図３９は、デバイスを取り付けているＵＳＤＭの好適例を示す略ブロック図である。
【図４０】図４０は、構成グループを示す略ブロック図である。
【図４１】図４１は、ホストインターフェースアーキテクチャを示す略ブロック図である。
【図４２】図４２は、Ｒバス読出し及び読込みサイクルを示す図である。
【図４３】図４３は、リアライザ設計変換システムを示す略ブロック図である。
【図４４】図４４は、本発明に用いられている設計部データ構造を示す図である。
【図４５】図４５は、本発明に用いられている基本要素変換を示す図である。
【図４６】図４６は、基本要素のクラスタへの移動を示す略ブロック図である。
【図４７】図４７は、シンプルな回路網相互接続を示す図である。
【図４８】図４８は、トライステート回路網相互接続を示す図である。
【図４９】図４９は、トライステート回路網相互接続を示す図である。
【図５０】図５０は、リアライザロジックシミュレーションシステムを示す略ブロック図である。
【図５１】図５１は、マルチステートロジックのリアライザシステム構成を示す略図である。
【図５２】図５２は、ディレイ依存機能の例を示す略図である。
【図５３】図５３は、ユニットディレイ構成の例を示す略図である。
【図５４】図５４は、リアルディレイ構成を示す略図である。
【図５５】図５５は、リアライザフォールトシミュレーションシステムを示す略ブロック図である。
【図５６】図５６は、リアライザロジックシミュレータ評価システムを示す略ブロック図である。
【図５７】図５７は、リアライザプロトタイピングシステムを示す略ブロック図である。
【図５８】図５８は、リアライザプロトタイピングシステムのデジタルコンピュータの一例を示す略ブロック図である。
【図５９】図５９は、仮想ロジックアナライザの構成を示す略ブロック図である。
【図６０】図６０は、リアライザ生産システムを示す略ブロック図である。
【図６１】図６１は、リアライザ計算システムを示す略ブロック図である。

【図６２】図６２は、ロジックボード、ボックス及びラックの階層相互接続を具える好適例の一般的なアーキテクチャを示す図である。
【図６３】図６３は、ロジックボード、ボックス及びラックの階層相互接続を具える好適例の一般的なアーキテクチャを示す図である。
【図６４】図６４は、ロジックボード、ボックス及びラックの階層相互接続を具える好適例の一般的なアーキテクチャを示す図である。
【図６５】図６５は、ロジックボード、ボックス及びＺレベルボックスの物理的な構成を示す図である。
【符号の説明】
１０…アレイ
１２…電気的に再構成可能なゲートアレイ
１４…基本要素コンバータ
１６…ホストＥＤＡシステム
１８…リアライザ設計変換システム
２０…テストベクトル
２２…ホストインタフェース
２４…刺激及び応答ベクトルメモリ
２６ａ…設計合成ツール
２６ｂ…ロジック合成ツール
２８…入力プログラム
３０…ホストインタフェース
３２…メモリモジュール
３４…クロスバーチップ
３６…多次元アレイ

【図１】

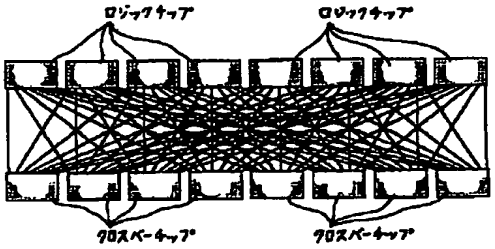


【図８】

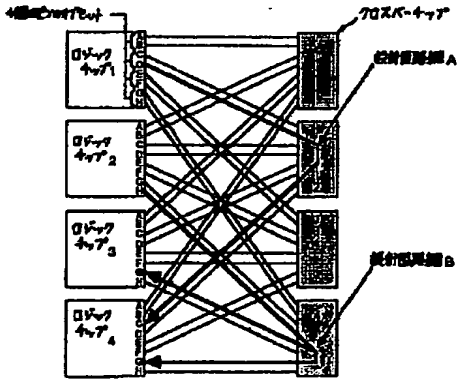


【図２０】

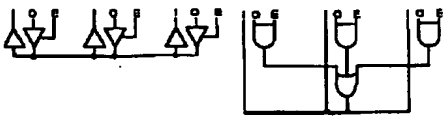
【図６】



【図７】

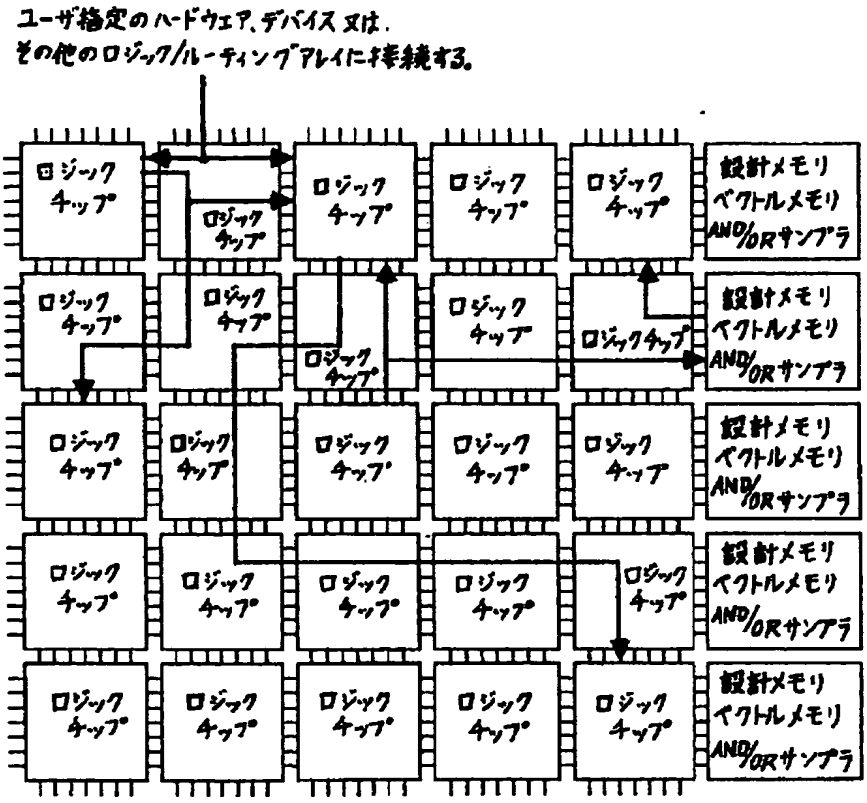


【図９】

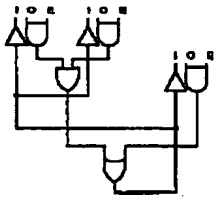


【図１０】

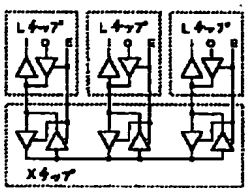
【図2】



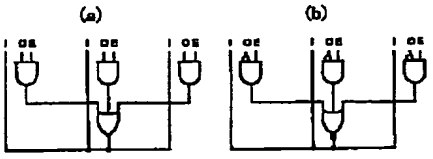
【図12】



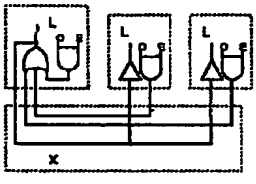
【図16】



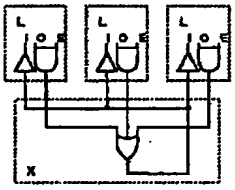
【図11】



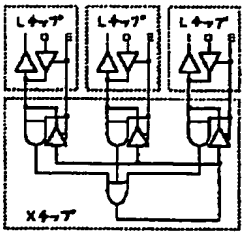
【図13】



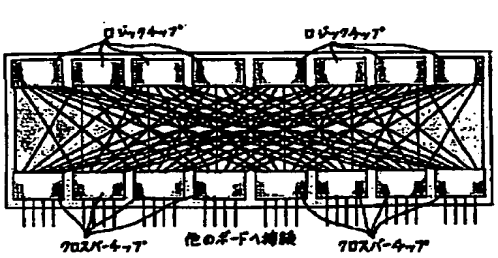
【図14】



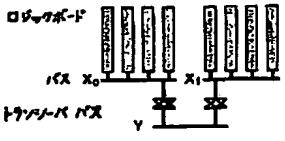
【図15】



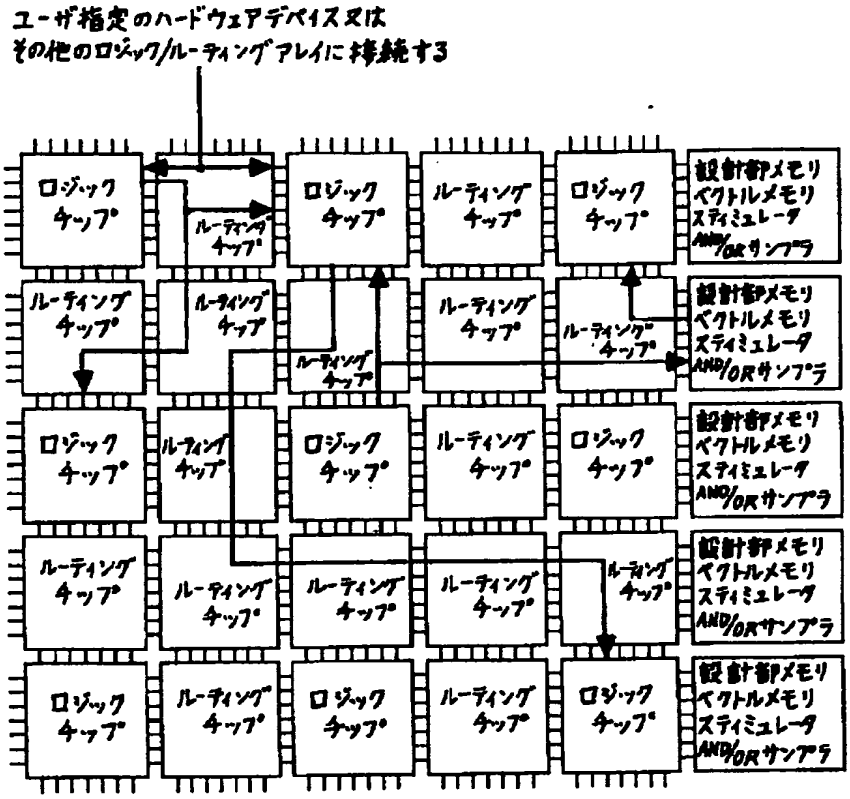
【図17】



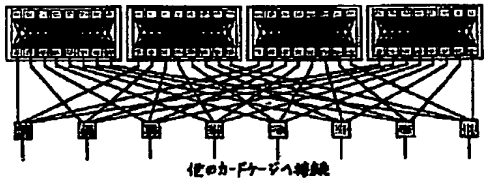
【図21】



【図3】

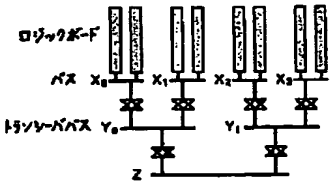


【図18】

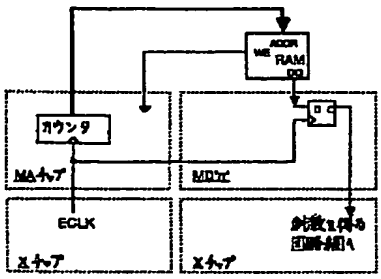


【図19】

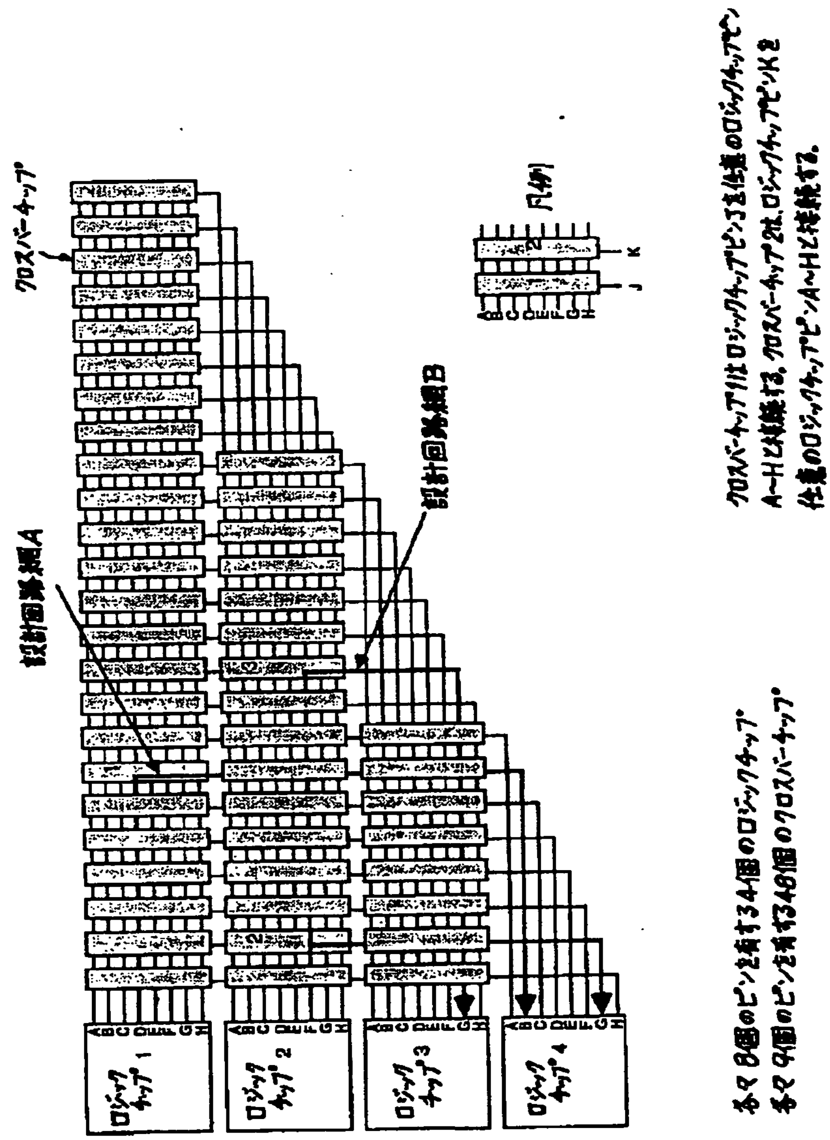
【図22】



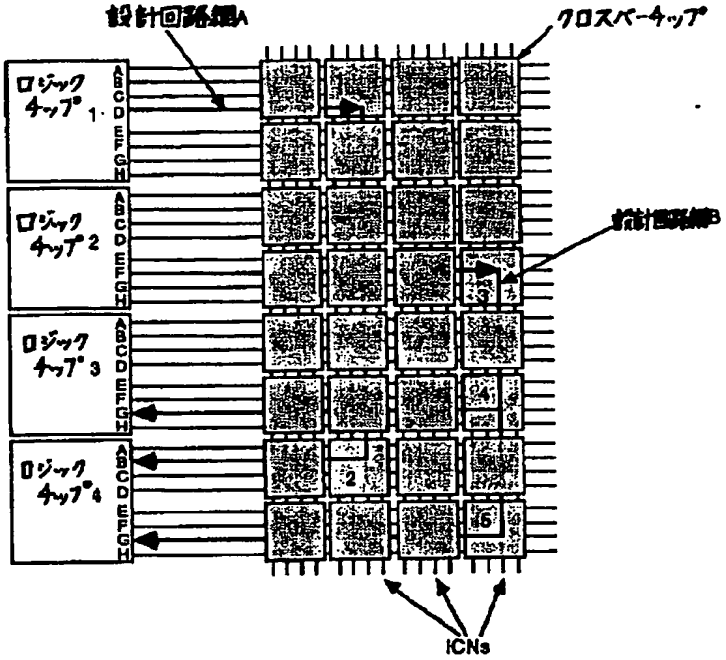
【図29】



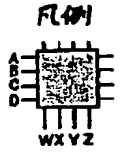
【図4】


クロスバーチップ
設計回路網A
設計回路網B
凡例
ロジックチップ1
ロジックチップ2
ロジックチップ3
ロジックチップ4
クロスバーチップ1はロジックチップピンJを任意のロジックチップピンA~Hと接続する。クロスバーチップ2はロジックチップピンKを任意のロジックチップピンA~Hと接続する。
各々8個のピンを有する4個のロジックチップ
各々9個のピンを有する48個のクロスバーチップ

【図5】

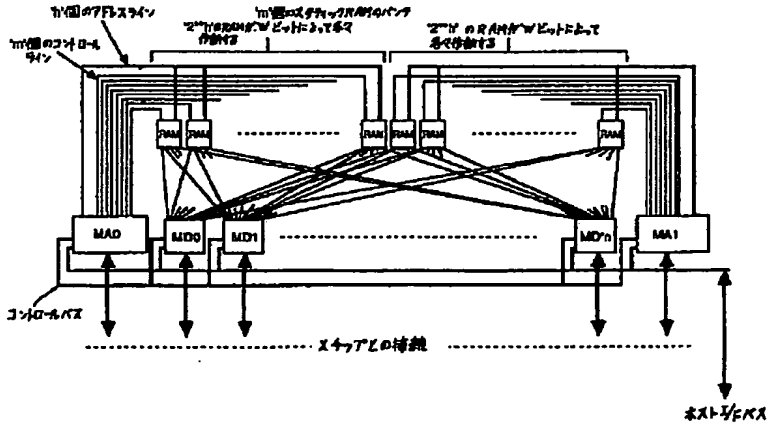


各々 8個のピンを有する4個のロジックチップ
各々 9個のピンを有する48個のクロスバーチップ

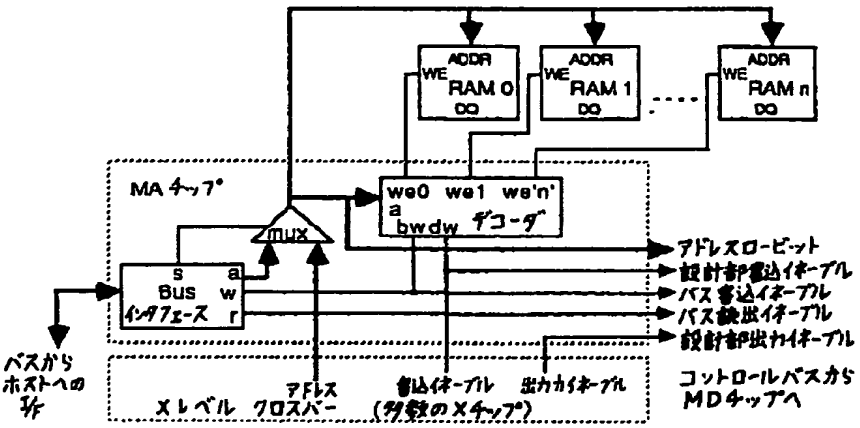


クロスバーチップ1は任意のロジックピンA~Dと、
任意のICN W~Zと接続する。

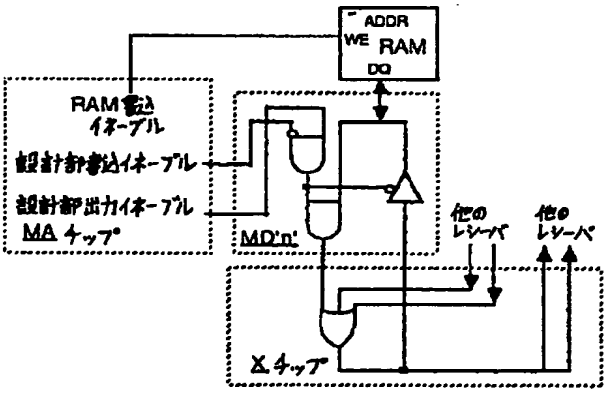
【図23】



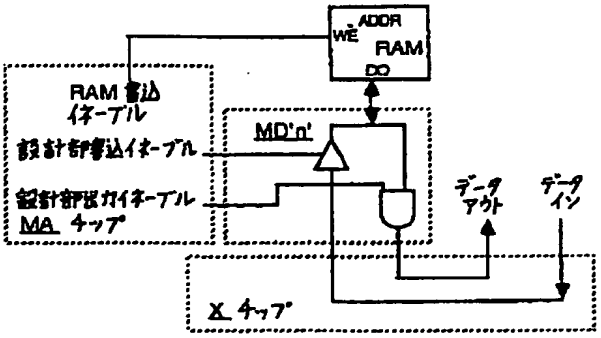
【図24】



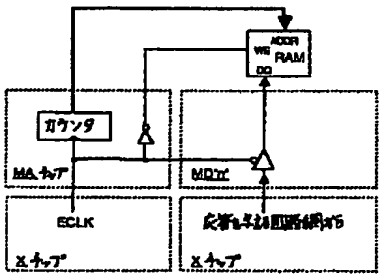
【図25】



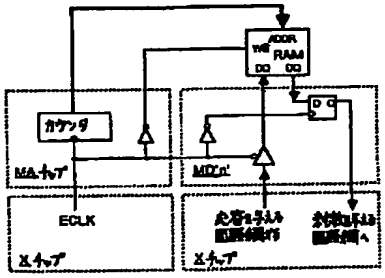
【図26】



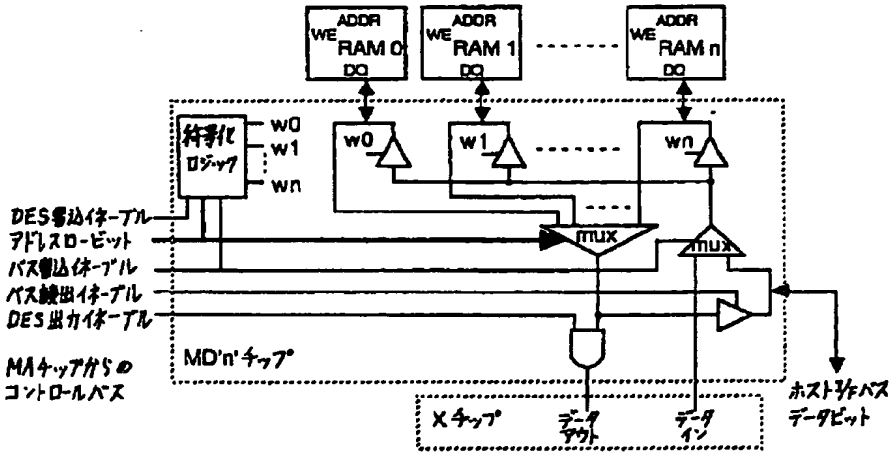
【図30】



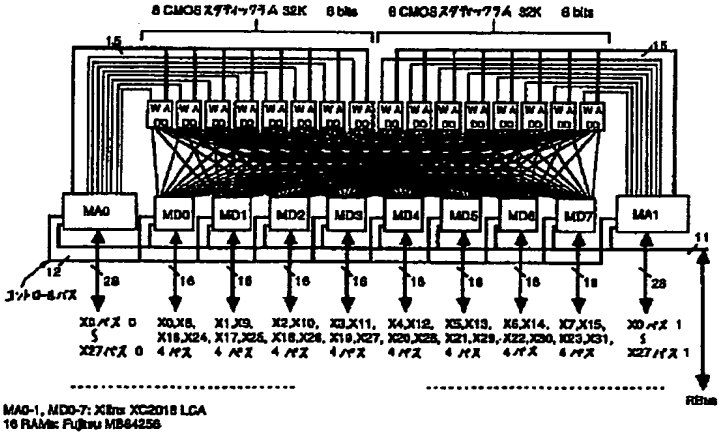
【図31】



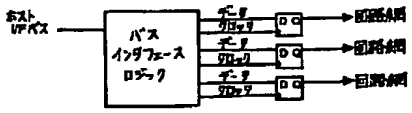
【図27】



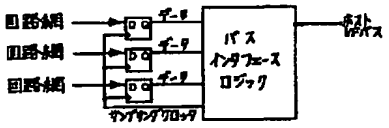
【図28】



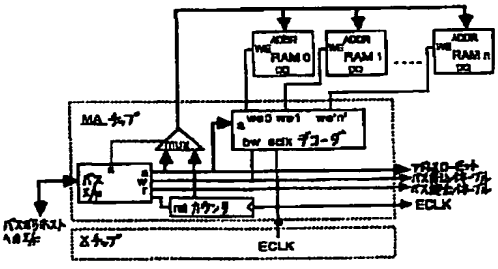
【図34】



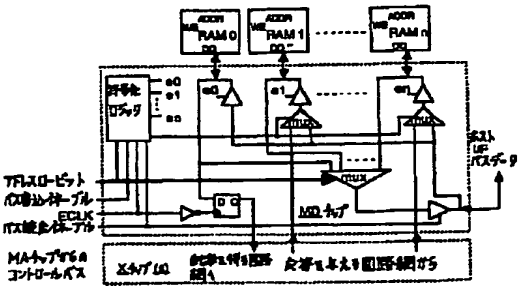
【図36】



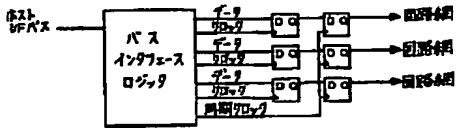
【図32】



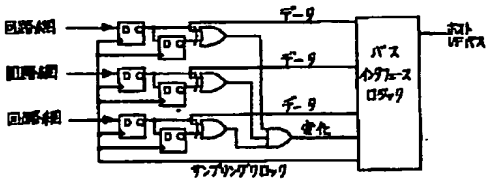
【図33】



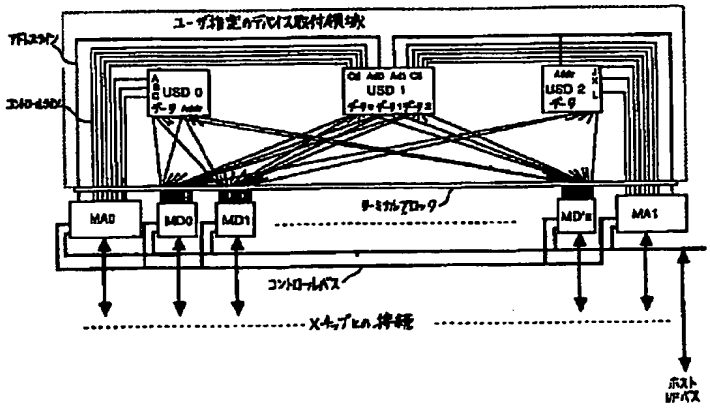
【図35】



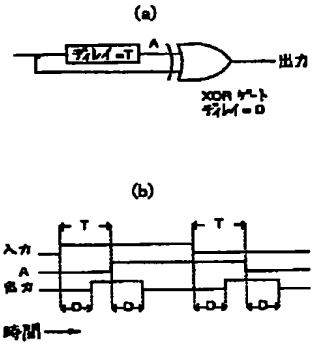
【図37】



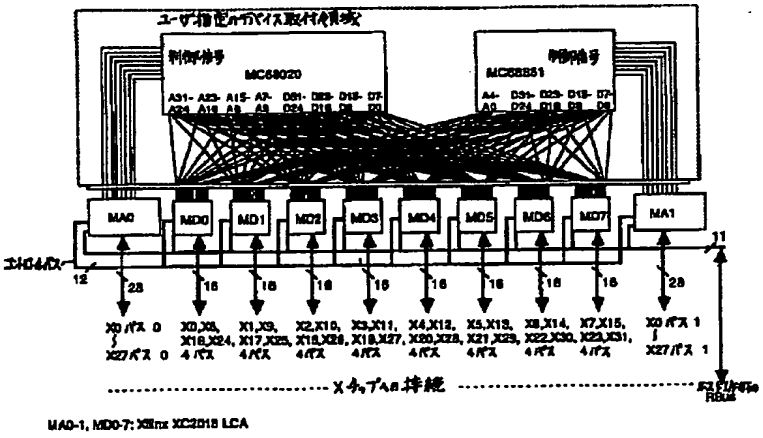
【図38】



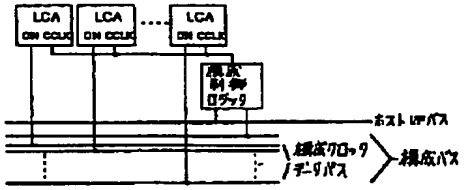
【図52】



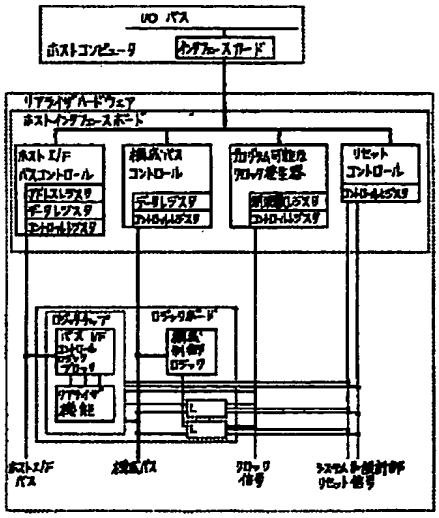
【図39】



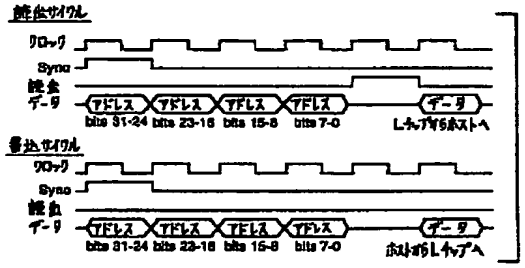
【図40】



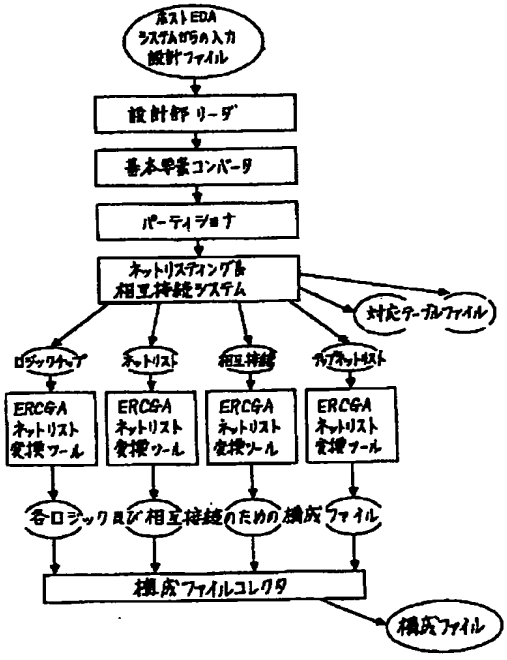
【図41】



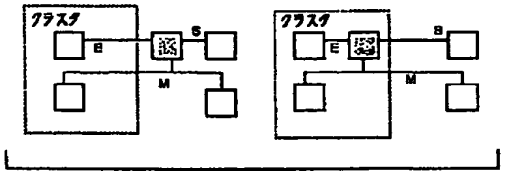
【図42】



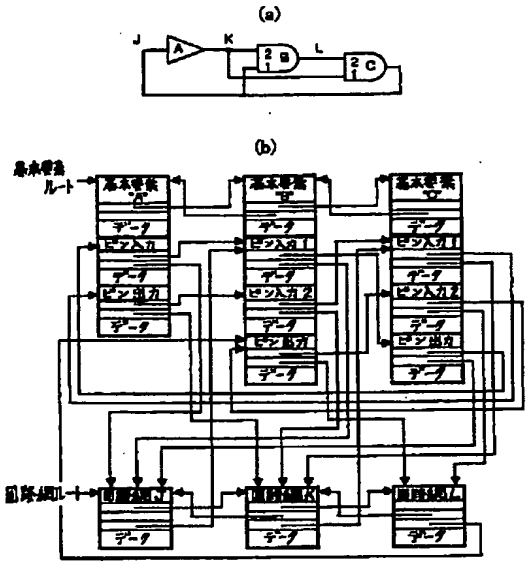
【図43】



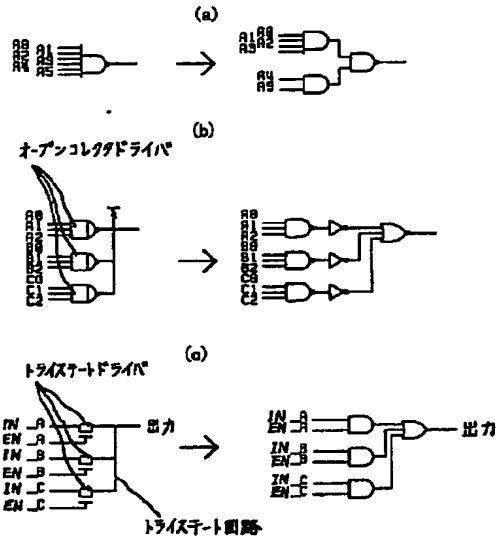
【図46】



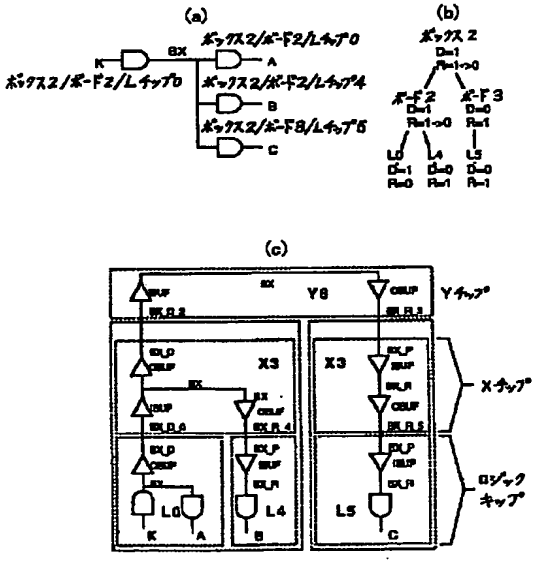
【図44】



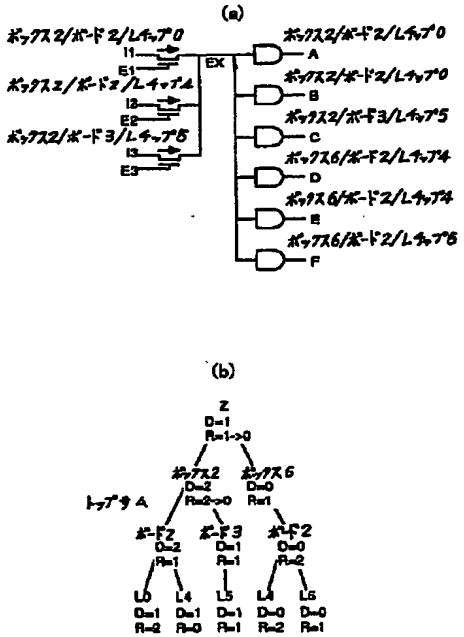
【図45】



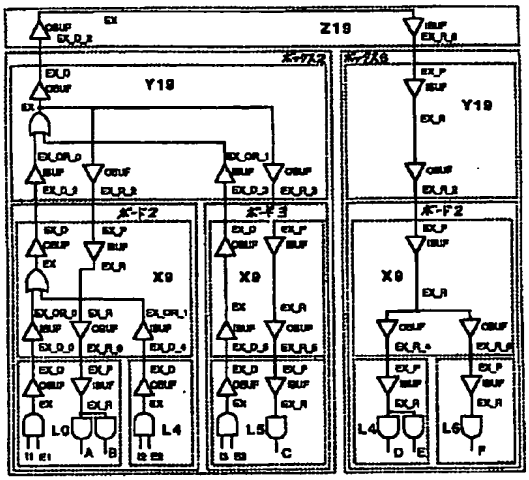
【図47】



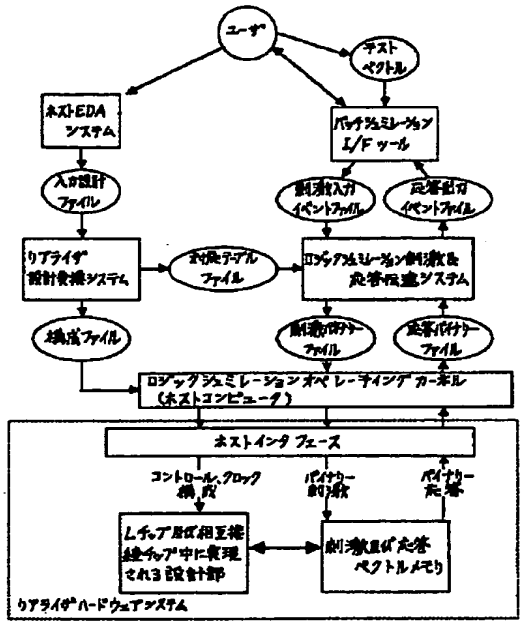
【図48】



【図49】



【図50】



【図51】

(a)

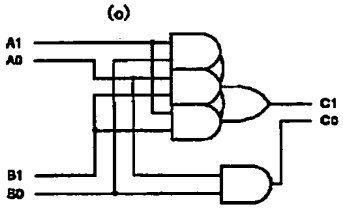
状態	bit 1	0
L	0	0
H	0	1
X	1	0

L＝ロー：0値
H＝ハイ：1値
X＝全値

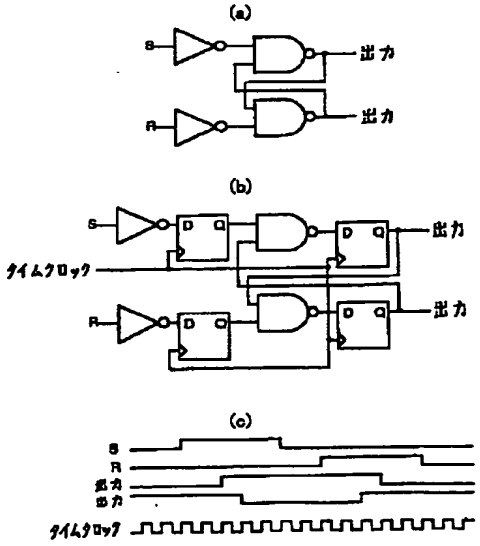
(b)

入力 A	B	A1	A0	B1	B0	出力 C	C1	C0
L	L	0	0	0	0	L	0	0
L	H	0	0	0	1	L	0	0
L	X	0	0	1	0	L	0	0
H	L	0	1	0	0	L	0	0
H	H	0	1	0	1	H	0	1
H	X	0	1	1	0	X	1	0
X	L	1	0	0	0	L	0	0
X	H	1	0	0	1	X	1	0
X	X	1	0	1	0	X	1	0

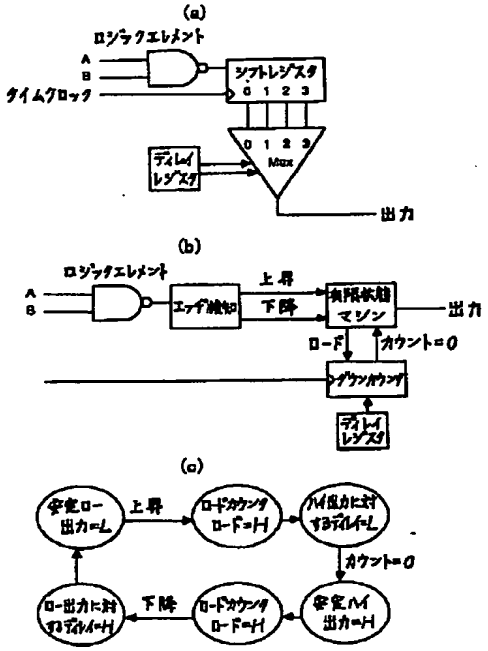
(c)



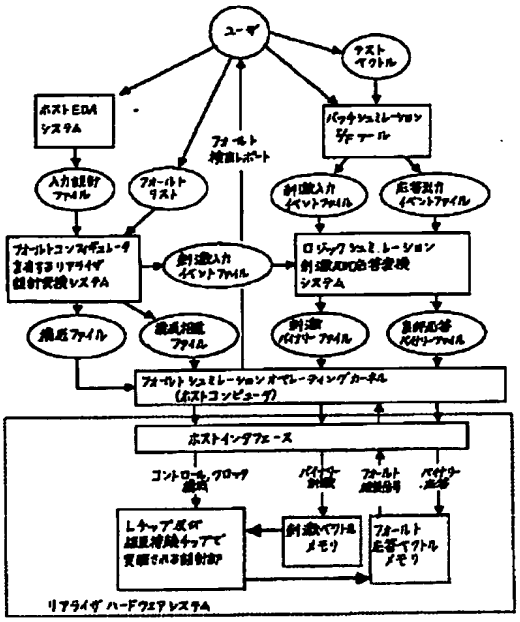
【図53】



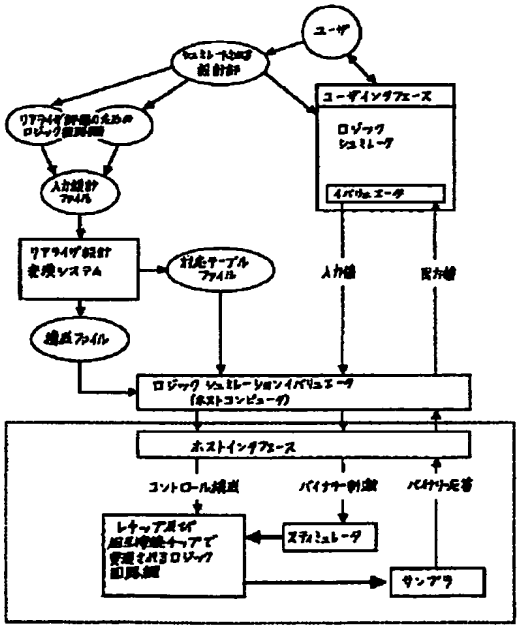
【図54】


(a)
ロジックエレメント
A
B
タイムクロック
シフトレジスタ
0 1 2 3
0 1 2 3
Mux
ディレイレジスタ
出力
(b)
ロジックエレメント
A
B
エッジ検知
上昇
下降
有限状態マシン
出力
ロード
カウント=0
ダウンカウンタ
ディレイレジスタ
(c)
安定ロー出力=L
上昇
ロードカウンタロード=H
ハイ出力に対するディレイ=L
カウント=0
安定ハイ出力=H
ロードカウンタロード=H
下降
ロー出力に対するディレイ=H

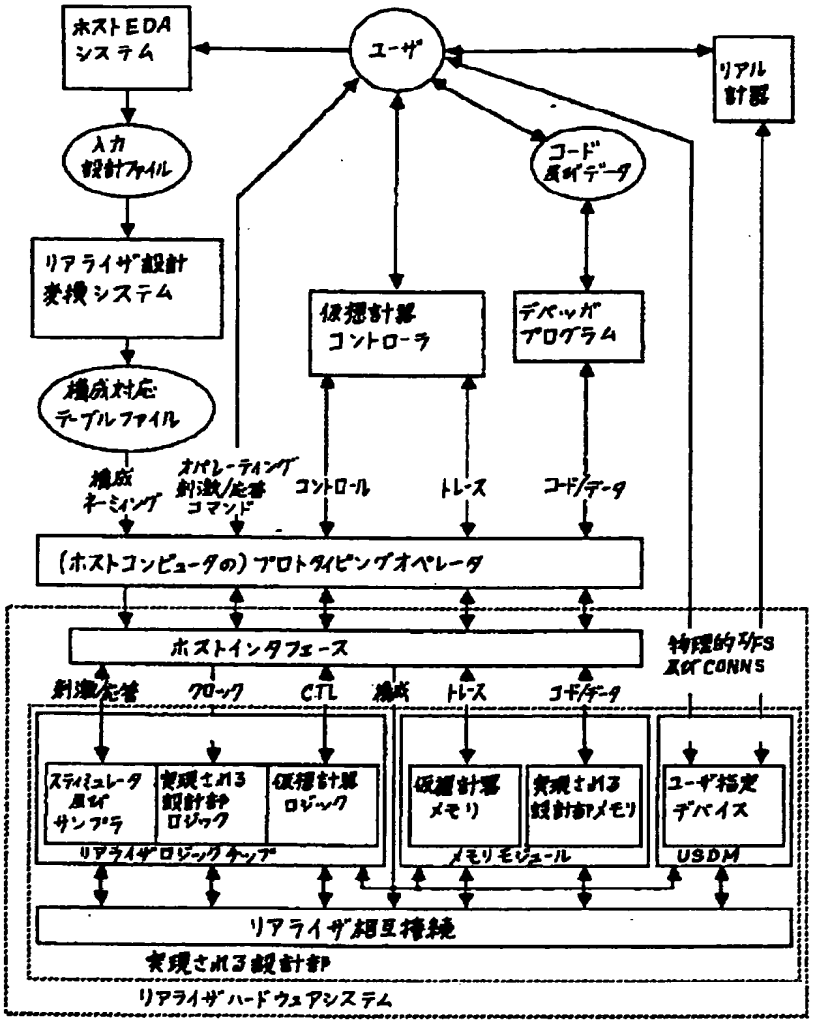
【図55】


ユーザ
テストベクトル
ポストEDAシステム
フォールト検出レポート
バックシュミレーションツール
入力波形ファイル
フォールトリスト
刺激入力イベントファイル
応答出力イベントファイル
フォールトコンパイラ
刺激入力イベントファイル
ロジックシュミレーション
ネットリスト
バイナリファイル
フォールトシュミレーションオペレーティングカーネル
(ホストコンピュータ)
ホストインタフェース
コントロール,クロック
バイナリ刺激
フォールト検出信号
バイナリ応答
レチャブルゲート
刺激ベクトルメモリ
フォールト応答ベクトルメモリ
リアライザハードウェアシステム

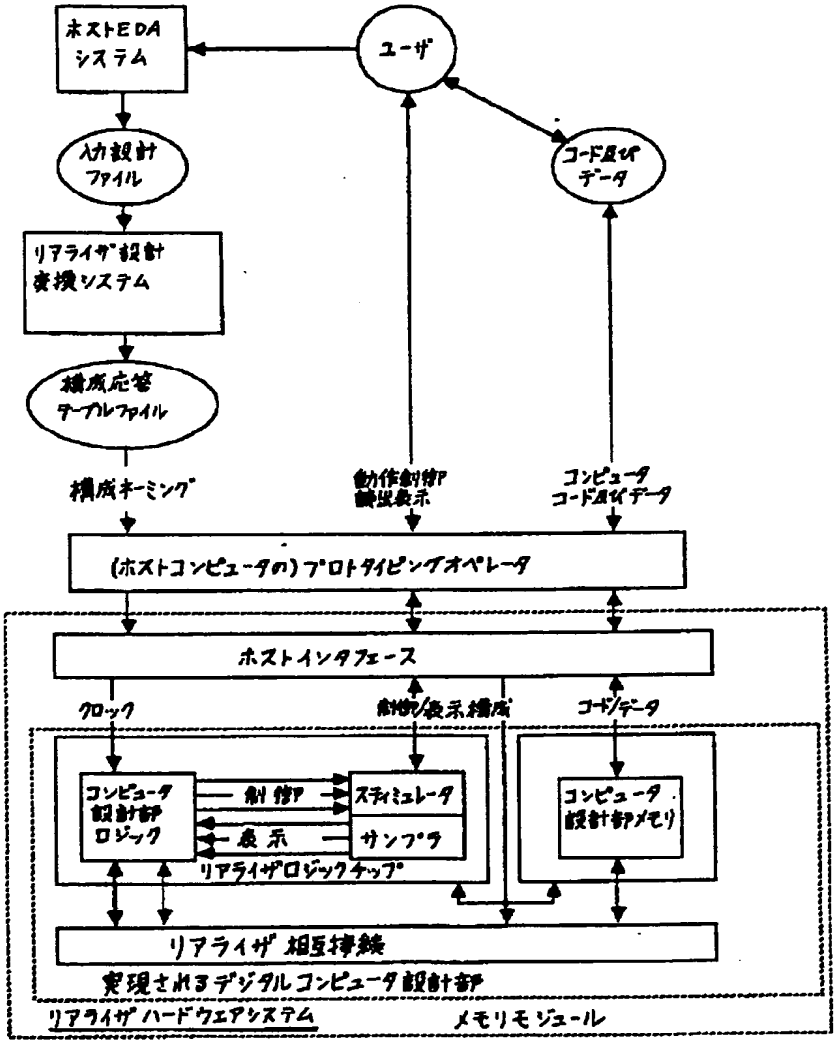
【図56】


ユーザ
ユーザインタフェース
ロジックシュミレータ
イベントエータ
入力波形ファイル
リアライザ設計変換システム
対応テーブルファイル
入力線
出力線
遅延ファイル
ロジックシュミレーションイベントエータ
(ホストコンピュータ)
ホストインタフェース
コントロール線
バイナリ刺激
バイナリ応答
スティミュレータ
サンプラ

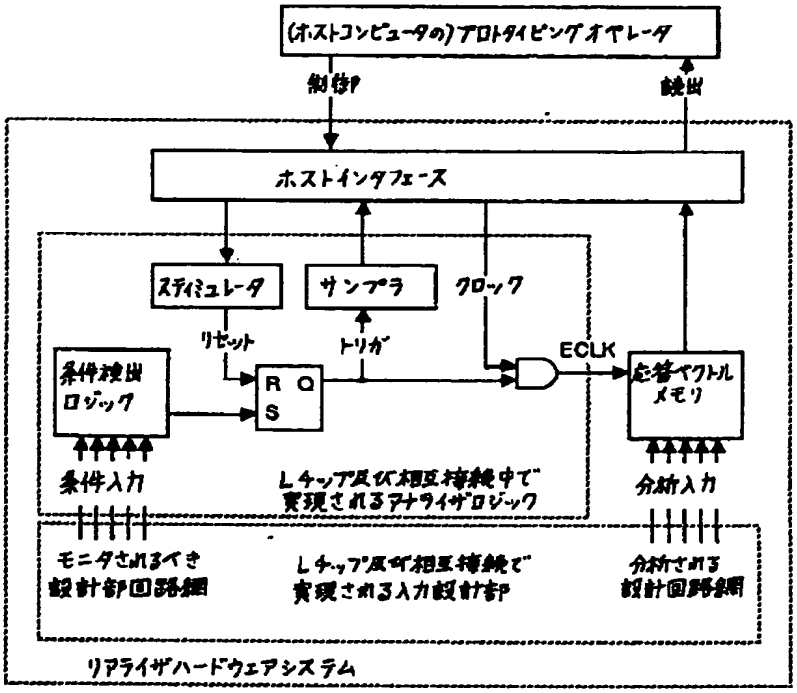
【図57】


ホストEDA システム
ユーザ
リアル 計器
入力 設計ファイル
コード 反射データ
リアライザ設計 変換システム
仮想計器 コントローラ
デバッガ プログラム
構成対応 テーブルファイル
構成 ネーミング
オペレーティング 刺激/応答 コマンド
コントロール
トレース
コード/データ
(ホストコンピュータの) プロトタイピングオペレータ
ホストインタフェース
物理的I/FS
刺激/応答
クロック
CTL
構成
トレース
コード/データ
スティミュレータ 反射 サンプラ
実現された設計部 ロジック
仮想計器 ロジック
リアライザロジックチップ
仮想計器 メモリ
実現された設計部メモリ
メモリモジュール
ユーザ指定 デバイス
USDM
リアライザ相互接続
実現された設計部
リアライザハードウェアシステム

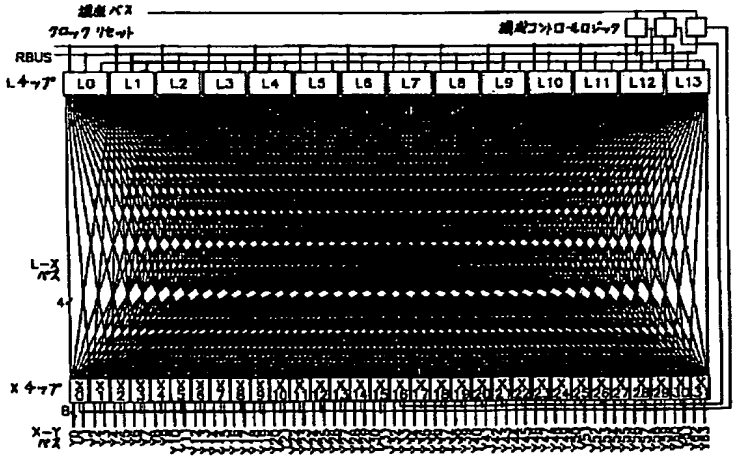
【図58】


ホストEDA システム
ユーザ
入力設計 ファイル
コード及び データ
リアライザ設計 支援システム
構成応答 テーブルファイル
構成ネーミング
動作制御P 論理表示
コンピュータ コードおよびデータ
(ホストコンピュータの) プロトタイピングオペレータ
ホストインタフェース
クロック
制御P/表示構成
コード/データ
コンピュータ 設計部 ロジック
制御P
表示
スティミュレータ
サンプラ
リアライザロジックチップ
コンピュータ 設計部メモリ
リアライザ 相互接続
実現されるデジタルコンピュータ設計部
リアライザ ハードウエアシステム
メモリモジュール

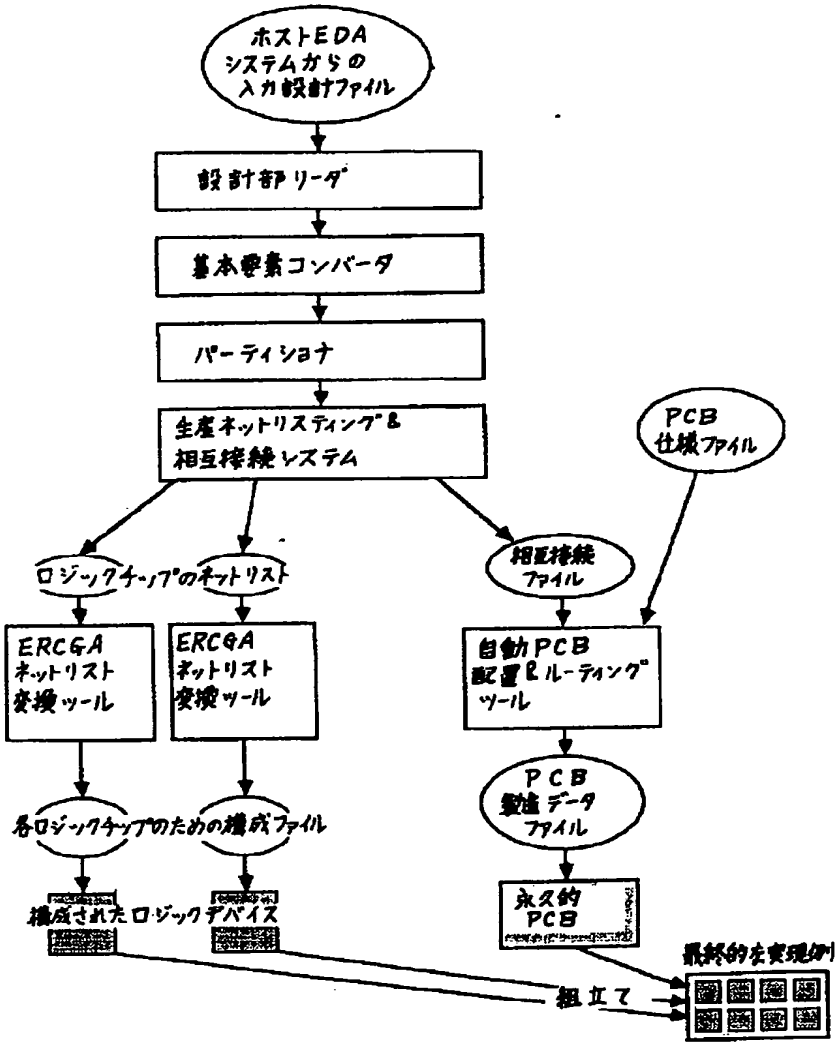
【図５９】


(ホストコンピュータの)プロトタイピングイマレータ
制御
読出
ホストインタフェース
スティミュレータ
サンプラ
クロック
リセット
トリガ
ECLK
条件検出ロジック
R Q S
応答ベクトルメモリ
条件入力
Lチップ及び相互接続中で実現されるアナライザロジック
分析入力
モニタされるべき設計部回路網
Lチップ及び相互接続で実現される入力設計部
分析される設計回路網
リアライザハードウェアシステム

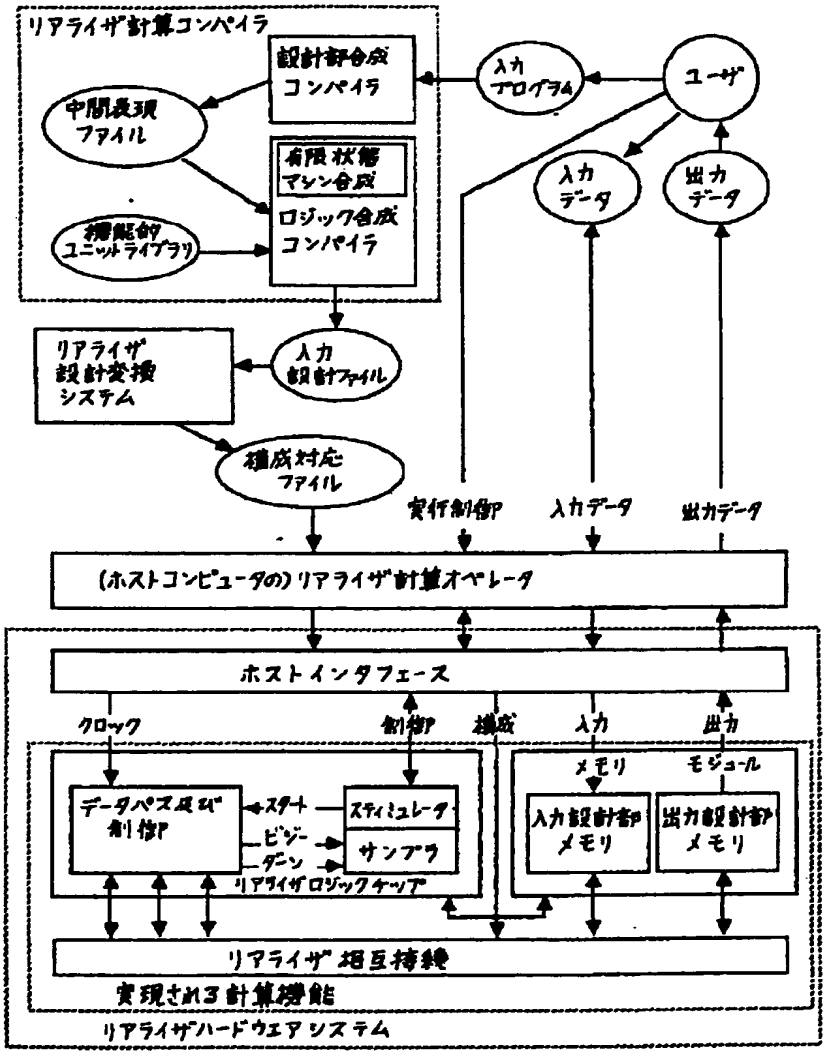
【図６２】


クロック リセット
RBUS
Lチップ
L0
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10
L11
L12
L13
Xチップ

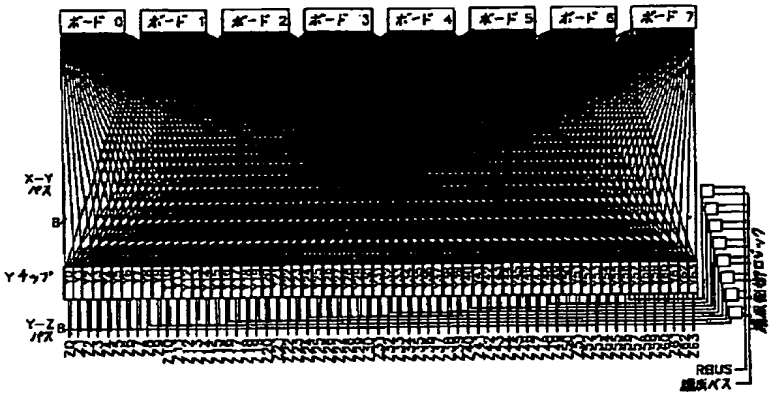
【図60】


ホストEDA
システムからの
入力設計ファイル
設計部リーダ
基本要素コンバータ
パーティショナ
生産ネットリスティング&
相互接続システム
PCB
仕様ファイル
ロジックチップのネットリスト
相互接続
ファイル
ERCGA
ネットリスト
変換ツール
ERCGA
ネットリスト
変換ツール
自動PCB
配置&ルーティング
ツール
各ロジックチップのための構成ファイル
PCB
製造データ
ファイル
構成されたロジックデバイス
永久的
PCB
組立て
最終的な実現例

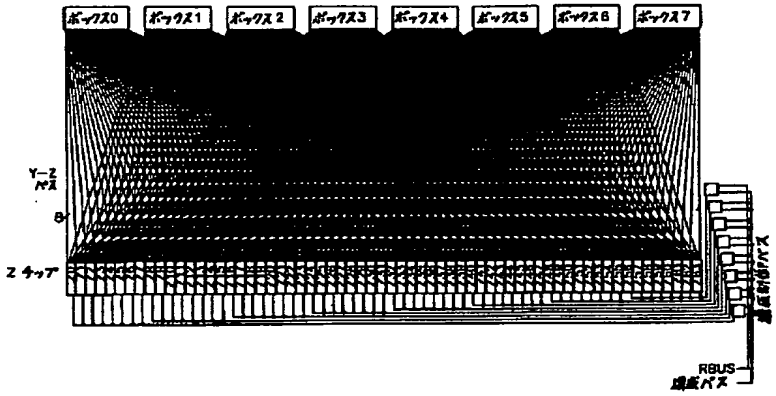
【図６１】


リアライザ計算コンパイラ
設計部合成コンパイラ
入力プログラム
ユーザ
中間表現ファイル
有限状態マシン合成
ロジック合成コンパイラ
入力データ
出力データ
機能的ユニットライブラリ
リアライザ設計変換システム
入力設計ファイル
構成対応ファイル
実行制御P
入力データ
出力データ
(ホストコンピュータの)リアライザ計算オペレータ
ホストインタフェース
クロック
制御P
構成
入力
出力
データパス及び制御P
スタート
ビジー
ダン
スティミュレータ
サンプラ
リアライザロジックチップ
メモリ
モジュール
入力設計部メモリ
出力設計部メモリ
リアライザ相互接続
実現される計算機能
リアライザハードウェアシステム

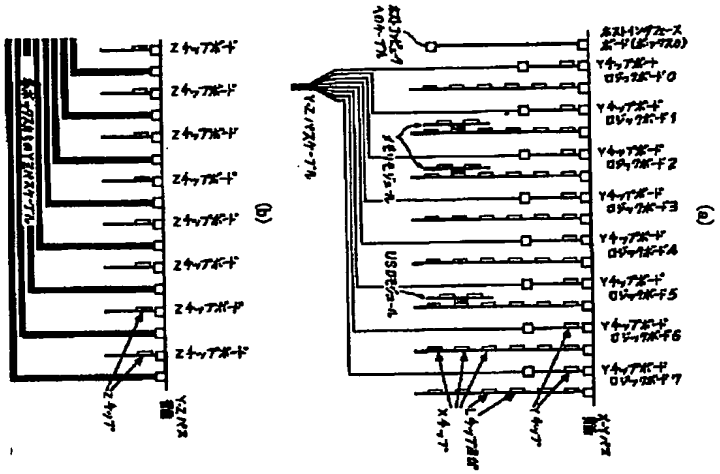
【図63】


ボード0
ボード1
ボード2
ボード3
ボード4
ボード5
ボード6
ボード7
X-Y バス
B
Yチップ
Y-Z バス
B
RBUS
結合バス

【図64】


ボックス0
ボックス1
ボックス2
ボックス3
ボックス4
ボックス5
ボックス6
ボックス7
Y-Z バス
B
Zチップ
RBUS
結合バス

【図65】



【手続補正書】
【提出日】平成１１年１２月２７日（１９９９．１２．２７）
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】発明の名称
【補正方法】変更
【補正内容】
【発明の名称】　電気的に再構成可能なゲートアレイを用いて論理構成を構築する方法
【手続補正２】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正内容】
【特許請求の範囲】
【請求項１】　複数のロジック回路素子及び前記ロジック回路素子間に接続経路を変更可能に確立する手段を有する電気的に再構成可能なゲートアレイ（ＥＲＣＧＡ）を複数用いて論理構成を構築する方法であって、
論理ゲートからなる複数の基本要素及び前記基本要素を相互接続する回路網を含む第１デジタルロジック回路網を表わす第１入力設計を供給する工程と、
前記第１入力設計によって表わされる前記第１のデジタルロジック回路網の前記基本要素及び相互接続する回路網を少なくとも第１及び第２区分に自動的に区分化する工程と、
前記第１入力設計の前記第１区分を第１ゲートアレイにプログラムし、前記第１入力設計によって表される前記第１デジタルロジック回路網の前記第１区分を第１ゲートアレイ上に構築する工程と、
前記第１入力設計の前記第２区分を第２ゲートアレイにプログラムし、前記第１入力設計によって表される前記第１デジタルロジック回路網の前記第２区分を第２ゲートアレイ上に構築する工程と、
前記第１及び第２ゲートアレイを相互接続手段を用いて再構成可能に相互接続し、前記第１入力設計によって表わされる前記相互接続する回路網を前記第１及び第２ゲートアレイ間に構築する工程と、
論理ゲートからなる複数の基本要素及び前記基本要素を相互接続する回路網を備えていることを除き、前記第１デジタルロック回路網と全く無関係である第２デジタルロジック回路網を表わす第２入力設計を供給する工程と、
前記第２入力設計によって表わされる前記第２デジタルロジック回路網を少なくとも第１及び第２区分に自動的に区分化する工程と、
前記第２入力設計の前記第１区分を前記第１ゲートアレイにプログラムし、前記第２入力設計によって表わされる前記第２デジタルロジック回路網の前記第１区分を前記第１ゲートアレイ上に構築する工程と、
前記第２入力設計の前記第２区分を前記第２ゲートアレイにプログラムし、前記第２入力設計によって表わされる前記第２デジタルロジック回路網の前記第２区分を前記第２ゲートアレイ上に構築する工程と、

前記第１及び第２ゲートアレイを相互接続手段を用いて再構成可能に相互接続し、前記第２入力設計によって表わされる前記相互接続する回路網を前記第１及び第２ゲートアレイ間に構築する工程とを含むことを特徴とする方法。

【請求項２】　複数のロジック回路素子及び前記ロジック回路素子間に接続経路を変更可能に確立する手段を含む電気的に再構成可能なゲートアレイ（ＥＲＣＧＡ）を複数用いて論理構成を構築する方法であって、(a)論理ゲートからなる複数の基本要素及び前記基本要素を相互接続する回路網を含む第１デジタルロジック回路網を表わす第１入力設計を供給する工程と、(b)前記第１入力設計によって表わされる前記第１デジタルロジック回路網の前記基本要素及び相互接続する回路網を少なくともＮ個の区分に区分化する工程と、(c) 前記第１入力設計の各区分を少なくともＮ個のゲートアレイにそれぞれプログラムし、前記第１入力設計によって表わされる前記第１デジタルロジック回路網の各区分を前記少なくともＮ個のゲートアレイ上に構築する工程と、(d) 前記少なくともＮ個のゲートアレイを相互接続手段を用いて再構成可能に相互接続し、前記第１入力設計によって表わされる前記相互接続する回路網が前記少なくともＮ個のゲートアレイ上に構築されるように、前記少なくともＮ個のゲートアレイのそれぞれを少なくとも１つの他のゲートアレイに接続する工程と、(e) 論理ゲートからなる複数の基本要素及び前記基本要素を相互接続する回路網を備えていることを除き、前記第１デジタルロック回路網と全く無関係である第２デジタルロジック回路網を表わす第２入力設計について前記工程(a)～(d)を繰り返す工程とを含むことを特徴とする方法。

【請求項３】　請求項１または２に記載の方法であって、前記再構成可能に相互接続する工程は、少なくとも一つの追加のゲートアレイによって行われることを特徴とする方法。

【請求項４】　請求項３に記載の方法であって、各ゲートアレイは複数のピンを有し、前記再構成可能に相互接続する工程は、前記少なくとも一つの追加のゲートアレイを、前記再構成可能なゲートアレイのピンのすべてではないが、少なくとも一つに接続することを特徴とする方法。

フロントページの続き

(51)Int.Cl.⁷	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
// Ｇ０６Ｆ　11/22	３３０	Ｈ０１Ｌ　21/82	Ａ

(72)発明者　バチェラー　ジョン　エイ
アメリカ合衆国オレゴン州　97132　ニューバーグ　ボックス　91　ルート　1